

DEVICE AND METHOD FOR CONTROLLING FA SYSTEM, AND CONTROL PROGRAM GENERATING METHOD

Patent number: JP9171405
Publication date: 1997-06-30
Inventor: KOYAMA MASAHIRO; MIYAKE NORIHISA
Applicant: HITACHI LTD
Classification:
- international: G05B19/418; G05B13/02; G05B15/02; G06F9/06
- european:
Application number: JP19950331666 19951220
Priority number(s): JP19950331666 19951220

[View INPADOC patent family](#)

Abstract of JP9171405

PROBLEM TO BE SOLVED: To describe control programs of FA system integrating into one, to automatically allocate it to respective processors and to convert the control program into programs for these processors. **SOLUTION:** The work specifications of respective cells consisting of the FA system are described in a single language, program module groups by functions are generated for performing sequence control, control of automatic machine and control of programmable peripheral equipment, and these respective program modules 1601 by functions are separated into program groups 1803 by processors composed of plural programs 1802 by processors. At this time, a unit block processing time estimating and evaluating means 1801d evaluates the processing time of each unit block corresponding to processor configuration and function data 1801c, and a unit block connecting means 1801e connects unit blocks 1602 so as to optimize the distribution of processing load at plural processors.

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平9-171405

(43)公開日 平成9年(1997)6月30日

(51)Int.Cl. ⁸	識別記号	序内整理番号	F I	技術表示箇所
G 0 5 B 19/418			G 0 5 B 19/417	A
13/02			13/02	J
15/02			G 0 6 F 9/06	5 3 0 V
G 0 6 F 9/06	5 3 0			5 3 0 S
		0360-3H	G 0 5 B 15/02	P
審査請求 未請求 請求項の数26 O L (全 38 頁)				

(21)出願番号 特願平7-331666

(22)出願日 平成7年(1995)12月20日

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72)発明者 小山 昌宏

茨城県土浦市神立町502番地 株式会社日立製作所機械研究所内

(72)発明者 三宅 徳久

茨城県土浦市神立町502番地 株式会社日立製作所機械研究所内

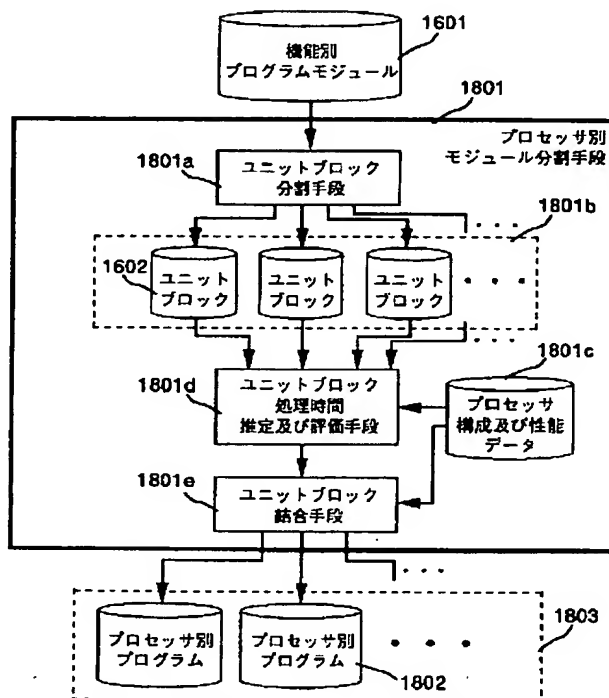
(74)代理人 弁理士 高崎 芳紘

(54)【発明の名称】 F Aシステムの制御装置と方法、制御プログラム生成方法

(57)【要約】

【課題】 F Aシステムの制御プログラムを一本化して記述し、自動的に各プロセッサに割り当て、そのプロセッサ用のプログラムに変換する。

【解決手段】 F Aシステムを構成する各セルの作業仕様を単一言語で記述し、シーケンス制御、自動機械の制御、プログラミング可能な周辺機器の制御を行う機能別プログラムモジュール群を生成し、これらの機能別プログラムモジュール1601の各々を複数のプロセッサ別プログラム1802からなるプロセッサ別プログラム群1803に分離する。この時、ユニットブロック処理時間推定及び評価手段1801dは、プロセッサ構成及び機能データ1801cにより各ユニットブロックの処理時間を評価し、複数のプロセッサにおける処理負荷の配分が最適になるように、ユニットブロック結合手段1801eがユニットブロック1602を結合する。



【特許請求の範囲】

【請求項1】 それぞれが、少なくとも自動機械と、前記自動機械に関連するプログラミング可能な周辺機器とを含む、複数のセルから構成されるFAシステムの制御装置であって、前記複数のセルに対して、それぞれ、前記セル全体としての作業仕様を同一言語で記述したセル制御プログラムと、

前記セル制御プログラムから、前記自動機械と周辺機器の作業順序の制御に関するプログラムと、前記自動機械の動作制御に関するプログラムと、前記プログラミング可能な周辺機器の制御に関するプログラムとに分離変換する分離変換手段と、

前記分離変換手段により分離変換された各プログラムに基づいて、当該セルを構成する前記自動機械と前記プログラミング可能な周辺機器の動作をそれぞれ制御する複数のプロセッサ手段と、を備えており、前記分離変換手段は、さらに、少なくとも前記自動機械あるいは前記プログラミング可能な周辺機器の動作をそれぞれ制御する複数の前記プロセッサ手段における処理負荷の配分を行う処理負荷配分手段を備えていることを特徴とするFAシステムの制御装置。

【請求項2】 前記請求項1に記載のFAシステムの制御装置において、前記分離変換手段の処理負荷配分手段は、さらに、前記分離変換されたプログラムをユニット毎に分解する手段と、前記ユニット毎に分解されたプログラムの前記プロセッサ手段による処理時間を評価する手段とを備えていることを特徴とするFAシステムの制御装置。

【請求項3】 前記請求項2に記載のFAシステムの制御装置において、前記分離変換手段の処理負荷配分手段は、さらに、当該セルを構成する前記自動機械あるいは前記プログラミング可能な周辺機器の動作をそれぞれ制御する複数のプロセッサ手段の構成あるいは性能に関するデータを提供する手段を備えていることを特徴とするFAシステムの制御装置。

【請求項4】 前記請求項3に記載のFAシステムの制御装置において、前記ユニット毎に分解されたプログラムの処理時間を評価する手段は、前記データ提供手段からのプロセッサ手段の構成あるいは性能に関するデータを基にして評価するように構成されていることを特徴とするFAシステムの制御装置。

【請求項5】 前記請求項4に記載のFAシステムの制御装置において、前記分離変換手段の処理負荷配分手段は、さらに、前記分解手段によりユニット毎に分解されたプログラムを、前記プログラムの処理時間評価手段による評価により結合する手段を備えていることを特徴とするFAシステムの制御装置。

【請求項6】 それぞれが少なくとも自動機械と、前記自動機械に関連するプログラミング可能な周辺機器とを含む、複数のセルから構成されるFAシステムを制御す

るための制御方法において、

前記複数のセルの少なくとも1のセルを構成する機器をそれぞれ複数のプロセッサにより制御するため、前記セル全体としての作業仕様を同一言語で記述してセル制御プログラムを作成し、当該作成したセル制御プログラムから、前記自動機械と周辺機器の作業順序の制御に関するプログラムと、前記自動機械の動作制御に関するプログラムと、前記プログラミング可能な周辺機器の制御に関するプログラムとに分離変換し、前記分離変換した各プログラムに基づいて、当該セルを構成する前記自動機械と前記プログラミング可能な周辺機器の動作をそれぞれ前記複数のプロセッサにより制御するFAシステムの制御方法であって、当該プログラムの分離変換の際に、さらに、少なくとも前記自動機械あるいは前記プログラミング可能な周辺機器の動作を制御するそれぞれの前記複数のプロセッサにおける処理負荷の配分を行うことを特徴とするFAシステムの制御方法。

【請求項7】 前記請求項6に記載のFAシステムの制御方法において、前記プロセッサにおける処理負荷の配分は、前記それぞれの機器の動作を制御する複数のプロセッサにおいて、特定のプロセッサに処理負荷が集中しないように分配されることを特徴とするFAシステムの制御方法。

【請求項8】 前記請求項6に記載のFAシステムの制御方法において、前記複数のプロセッサにおける処理負荷の配分は、前記分離変換されたプログラムをユニット毎に分解し、前記それぞれの機器の制御を行う前記複数のプロセッサの構成あるいは性能に基づいて前記ユニット毎に分解されたプログラムの各々を評価し、当該評価の結果に基づいて、前記各プロセッサ毎に結合することにより行うことを特徴とするFAシステムの制御方法。

【請求項9】 それぞれが、少なくとも自動機械と、前記自動機械に関連するプログラミング可能な周辺機器とを含む、複数のセルから構成されるFAシステムにおいて、前記複数のセルの少なくとも1のセルを構成する機器をそれぞれ複数のプロセッサにより制御するための制御プログラムを生成するためのFAシステムの制御プログラム生成方法であって、

前記セル全体としての作業仕様を同一言語で記述してセル制御プログラムを作成し、

当該作成したセル制御プログラムから、前記自動機械と周辺機器の作業順序の制御に関するプログラムと、前記自動機械の動作制御に関するプログラムと、前記プログラミング可能な周辺機器の制御に関するプログラムとに自動的に分離変換し、そして、

少なくとも前記自動機械あるいは前記プログラミング可能な周辺機器の動作を制御するそれぞれの前記複数のプロセッサにおける処理負荷の配分を自動的に行う、ことを特徴とするFAシステムの制御プログラム生成方法。

【請求項10】 前記請求項9に記載のFAシステムの

制御プログラム生成方法において、前記複数のプロセッサにおける処理負荷の配分は、前記分離変換されたプログラムをユニット毎に分解し、前記それぞれの機器の制御を行う前記複数のプロセッサの構成あるいは性能に基づいて前記ユニット毎に分解されたプログラムの各々を評価し、そして、当該評価の結果に基づいて、前記各プロセッサ毎に結合することにより行うことを特徴とするFAシステムの制御プログラム生成方法。

【請求項11】 ロボットを含む自動機械と視覚センサなどのプログラミング可能な周辺機器とを含む複数機器の単位（セル）から構成されるFAシステムの制御方法において、

前記セル全体としての作業仕様を直接記述し、前記複数機器の作業順序の制御に関する情報と、前記複数機器の入出力の制御に関する情報と、前記自動機械の動作制御に関する情報と、前記プログラミング可能な周辺機器の制御に関する情報と、前記複数機器間の同期をとるための動作タイミングに関する情報とを有するセル制御プログラムから、セル制御プログラム変換手段によって、前記複数機器の作業順序に関する情報と、前記入出力の制御に関する情報を分離及び抽出して、前記複数機器の作業順序の制御と前記入出力の制御を行う処理が記述されたシーケンス制御プログラムを生成し、また、前記自動機械の動作制御と前記動作タイミングに関する情報を分離及び抽出して、前記自動機械の位置決め及び軌道の制御を行う処理が記述された自動機械制御プログラムを生成し、また、前記プログラミング可能な周辺機器の制御と前記動作タイミングに関する情報を分離及び抽出して、前記プログラミング可能な周辺機器の制御を行う処理が記述された周辺機器制御プログラムを生成し、このシーケンス制御プログラムと自動機械制御プログラムと周辺機器制御プログラムとを、それぞれシーケンス制御プログラム解釈実行手段と自動機械制御プログラム解釈実行手段と周辺機器制御プログラム解釈実行手段とに出力することを特徴とするFAシステムの制御方法。

【請求項12】 ロボットなどの自動機械と視覚センサなどのプログラミング可能な周辺機器とを含む複数機器の単位（セル）から構成されるFAシステムの制御方法において、

前記セル全体としての作業仕様を直接記述し、前記複数機器の作業順序の制御に関する情報と、前記複数機器の入出力の制御に関する情報と、前記自動機械の動作制御に関する情報と、前記プログラミング可能な周辺機器の制御に関する情報と、前記複数機器間の同期をとるための動作タイミングに関する情報とを有するセル制御プログラムから、機能別モジュール分離変換手段によって、前記複数機器の作業順序に関する情報と、前記入出力の制御に関する情報を分離及び抽出して、前記複数機器の作業順序の制御と前記入出力の制御を行う処理が記述されたシーケンス制御プログラムモジュールを生成し、ま

た、前記自動機械の動作制御と前記動作タイミングに関する情報を分離及び抽出して、前記自動機械の位置決め及び軌道の制御を行う処理が記述された自動機械制御プログラムモジュールを生成し、また、前記プログラミング可能な周辺機器の制御と前記動作タイミングに関する情報を分離及び抽出して、前記プログラミング可能な周辺機器の制御を行う処理が記述された周辺機器制御プログラムモジュールを生成し、さらにこのシーケンス制御プログラムモジュールと自動機械制御プログラムモジュールと周辺機器制御プログラムモジュールとを、それぞれシーケンス制御プロセッサ別モジュール分割手段と自動機械制御プロセッサ別モジュール分割手段と周辺機器制御プロセッサ別モジュール分割手段によって、前記セルを制御するプロセッサの構成と性能に基づいて、各プロセッサの処理負荷の配分が最適となるようなプログラムモジュールに分割し、さらにこれらの分割したプログラムモジュールを各プロセッサが実行可能な形式のプログラムに変換して、シーケンス制御プロセッサ用プログラム群と自動制御プロセッサ用プログラム群と周辺装置制御プロセッサ用プログラム群とを生成し、これらの各プロセッサ用プログラム群を、それぞれを解釈実行するプロセッサ群に対して出力することを特徴とするFAシステムの制御方法。

【請求項13】 前記セル制御プログラムは、ベトリネットによって前記複数機器の動作状態の遷移を表現し、これに前記複数機器の動作内容を定義するコマンド列と、前記動作状態の終了を定義する条件式とを付加したプログラムであることを特徴とする請求項12記載のFAシステムの制御方法。

【請求項14】 前記セル制御プログラムは、前記ベトリネットを編集するウィンドウと、前記コマンド列を編集するウィンドウと、前記条件式を編集するウィンドウとを含むグラフィカル・ユーザ・インタフェースを備えたセル制御プログラム編集手段によって作成することを特徴とする請求項13記載のFAシステムの制御方法。

【請求項15】 前記セル制御プログラムは、前記ベトリネットの構造と一対一に対応するプログラムコードとして表現し、記憶することを特徴とする請求項13記載のFAシステムの制御方法。

【請求項16】 前記シーケンス制御プログラムモジュールと前記自動機械制御プログラムモジュールと前記周辺機器制御プログラムモジュールは、前記複数機器の動作タイミングの同期をとるための情報として、前記動作状態の整理番号を示す命令コードを含むことを特徴とする請求項13記載のFAシステムの制御方法。

【請求項17】 前記セル制御プログラムは、前記複数機器のうち、あるまとまった機器群で、その機器群どうしが互いに並行動作可能なもの（ユニット）の動作シーケンスをベトリネットの一つのモジュールとして記述し、これらのモジュールどうしを同期をとるためのプレ

ースによって結合することを特徴とする請求項13記載のFAシステムの制御方法。

【請求項18】 前記セル制御プログラムは、前記モジュールを一つのプログラムブロック（ユニットブロック）とした構造を有するプログラムコードとして表現することを特徴とする請求項17記載のFAシステムの制御方法。

【請求項19】 前記シーケンス制御プロセッサ別モジュール分割手段と前記自動機械制御プロセッサ別モジュール分割手段と前記周辺機器制御プロセッサ別モジュール分割手段は、前記ユニットブロックを単位としてプログラムモジュールの分割を行うことを特徴とする請求項18記載のFAシステムの制御方法。

【請求項20】 前記シーケンス制御プロセッサ別モジュール分割手段と前記自動機械制御プロセッサ別モジュール分割手段と前記周辺機器制御プロセッサ別モジュール分割手段は、前記ユニットブロックの対象とするプロセッサによる処理時間を推定し、その処理時間が制御上必要とされる基準を満たす範囲で前記ユニットブロックを結合し、前記シーケンス制御プロセッサ用プログラム群と前記自動制御プロセッサ用プログラム群と前記周辺装置制御プロセッサ用プログラム群とを生成することを特徴とする請求項19記載のFAシステムの制御方法。

【請求項21】 前記シーケンス制御プロセッサ用プログラム群と前記自動制御プロセッサ用プログラム群と前記周辺装置制御プロセッサ用プログラム群は、共有バスを介して結合されるシーケンス制御プロセッサ群と自動機械制御プロセッサ群と周辺機器制御プロセッサ群によって解釈実行することを特徴とする請求項12記載のFAシステムの制御方法。

【請求項22】 前記シーケンス制御プロセッサ群と前記自動機械制御プロセッサ群と前記周辺機器制御プロセッサ群とは、前記共有バスに接続された共有データ記憶手段に格納される前記複数機器の動作タイミングの同期をとるための情報にアクセス可能であることを特徴とする請求項21記載のFAシステムの制御方法。

【請求項23】 前記シーケンス制御プロセッサ用プログラム群と前記自動制御プロセッサ用プログラム群と前記周辺装置制御プロセッサ用プログラム群は、それぞれネットワークで接続された複数のプログラマブルコントローラと自動機械コントローラと周辺機器コントローラであることを特徴とする請求項12記載のFAシステムの制御方法。

【請求項24】 前記プログラマブルコントローラと前記自動機械コントローラと前記周辺機器コントローラは、前記ネットワークを介して、前記複数機器の動作タイミングの同期をとるための情報を互いに交換し、共有し合うことを特徴とする請求項23記載のFAシステムの制御方法。

【請求項25】 前記シーケンス制御プロセッサ用プロ

グラム群と前記自動制御プロセッサ用プログラム群と前記周辺装置制御プロセッサ用プログラム群は、共有バスを介して結合されるシーケンス制御プロセッサ群と自動機械制御プロセッサ群と周辺機器制御プロセッサ群とから構成されるマルチプロセッサ型コントローラを、ネットワークによって複数個接続したものであることを特徴とする請求項12記載のFAシステムの制御方法。

【請求項26】 前記マルチプロセッサ型コントローラは、前記ネットワークを介して、前記複数機器の動作タイミングの同期をとるための情報を互いに交換し、共有し合うことを特徴とする請求項25記載のFAシステムの制御方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ロボットを含むFAシステムの制御装置と方法に係り、特に、ロボットなどの自動機械と視覚センサなどのプログラミング可能な周辺機器を含む複数機器によって構成されるFAシステムの制御装置と方法、さらには、かかるシステムにおける制御プログラムの生成方法に関する。

【0002】

【従来の技術】ロボットなどの自動機械と、視覚センサなどのプログラミング可能な周辺機器とを含む複数の機器によって構成されるFA（Factory Automation）システムは、既に広く知られ、かつ、実用化されている。そして、かかる従来のFAシステム及びその制御方法は、一般に、各種機器の入出力（以下、「I/O」と略記）を制御するプログラマブルコントローラと、ロボットを制御するロボットコントローラと、画像処理装置などを、パラレルI/O、または、ネットワークによって接続し、FAシステムを構成する各機器の制御を行うというものであった。そして、かかる従来技術になるFAシステムの制御においては、各機器の制御対象毎に異なる制御装置（例えば、プログラマブルコントローラ、ロボットコントローラ、画像処理装置など）を組み合わせ使用し、しかも、制御装置毎に異なるプログラミング言語を使用していた。

【0003】このため、FAシステムを構築する場合、幾つもの異なったプログラム言語を習得する必要があり、一つのFAシステムを制御するためには、このような異なったプログラム言語を駆使して、互に関連し合う複数のプログラムを作成しなければならなかった。また、FAシステムの機能が高度化するにつれて、FAシステムの制御装置の構成も複雑となり、そのプログラムの作成には多大な労力を要し、その際にも、上記異なるプログラミング言語の使用が、やはり、FAシステムを制御するプログラムの開発効率を向上させる上で、大きな阻害要因となっていた。

【0004】ところで、従来、上記のような課題を解決するものとして、例えば、特開平7-84768号公報

によれば、F A用機器を作動させるプログラムを一つに統合した形で作成するF Aシステムの制御方法が記載されている。また、やはり上記のような課題を解決するため、例えば、特開平7-72920号公報によれば、F Aシステムの制御プログラムを一括して記述するF Aシステムの制御方法が既に提案されている。

【0005】すなわち、上記従来技術の前者である特開平7-84768号公報に記載されているF Aシステムの制御方法は、ロボットや画像処理装置などのF A用機器を一つのプログラミング言語で統一的に作動させるための統合プログラムを作成して、これをプログラマブルコントローラ、ロボットコントローラ、画像処理装置で実行されるプログラムに分解するものである。なお、ここで述べられている統合プログラムは、従来のプログラマブルコントローラのラダープログラムに、ロボットプログラムと画像処理プログラムをサブルーチンのような形で付加した、いわゆる拡張型ラダープログラムなどであり、これをラダープログラム、ロボットプログラム、画像処理プログラムの三つに分解する方法をとっている。

【0006】また、上記後者の従来技術である特開平7-72920号公報に記載されているF Aシステムの制御方法は、F Aシステムの作業を一括して記述した制御プログラムから、シーケンス制御プログラムとロボット制御プログラムを自動的に生成するものである。ここで述べられているF Aシステムの制御プログラムは、ベトリネットを応用したプログラミング言語により、F Aシステム全体としての作業仕様を明示的に表現したものである。

【0007】

【発明が解決しようとする課題】しかしながら、上記従来技術により提案されるF Aシステムの制御方法や制御プログラムは、F Aシステムの機能の高度化に伴うプログラムの開発の効率を向上させる上ではその効果を発揮するものの、以下に指摘する問題点については言及されていない。

【0008】すなわち、F Aシステムの機能が高度化するに伴い、複数の制御装置または制御用プロセッサによる分散協調化が進んだ場合、上記二つの従来技術により提案されるF Aシステムの制御方法や制御プログラムのように、単に、プログラムをシーケンス制御やロボット制御などのような機能別に分離するだけでなく、さらに、制御装置や制御プロセッサの構成と性能に応じた最適な処理負荷の配分になるように、各制御装置毎またはプロセッサ毎のプログラムの分割を行う必要性が生じる。例えば、シーケンス制御プロセッサ、ロボット制御プロセッサ、画像処理プロセッサのそれぞれが複数個組み合わされたような制御システムにおいては、各プロセッサに対する最適なプログラム分割を自動的に行うことができれば、複雑なF Aシステムを構築する際にも、ユ

ーザは制御システムのハードウェア構成を意識することなく、より効率のよいF Aシステムの制御プログラムを一本化して作成することができることとなる。

【0009】そこで、本発明では、上記の従来技術における問題点に鑑み、さらには、上記発明者等による従来技術に対する認識に基づいて、すなわち、F Aシステムの制御プログラムを一本化して作成することができるだけでなく、さらに、作成される制御プログラムによりF Aシステムをより効率的に制御することを可能にするF Aシステムの制御装置と方法、さらには、かかるシステムにおける制御プログラムの生成方法を提供することを目的とする。

【0010】

【課題を解決するための手段】上記の本発明の目的を達成するため、本発明によれば、それぞれが、少なくとも自動機械と、前記自動機械に関連するプログラミング可能な周辺機器とを含む、複数のセルから構成されるF Aシステムの制御装置であって、前記複数のセルに対して、それぞれ、前記セル全体としての作業仕様を同一言語で記述したセル制御プログラムと、前記セル制御プログラムから、前記自動機械と周辺機器の作業順序の制御に関するプログラムと、前記自動機械の動作制御に関するプログラムと、前記プログラミング可能な周辺機器の制御に関するプログラムとに分離変換する分離変換手段と、前記分離変換手段により分離変換された各プログラムに基づいて、当該セルを構成する前記自動機械と前記プログラミング可能な周辺機器の動作をそれぞれ制御する複数のプロセッサ手段とを備えており、前記分離変換手段は、さらに、少なくとも前記自動機械あるいは前記プログラミング可能な周辺機器の動作をそれぞれ制御する複数の前記プロセッサ手段における処理負荷の配分を行う処理負荷配分手段を備えているF Aシステムの制御装置が提起されている。

【0011】また、本発明によれば、やはり上記の本発明の目的を達成するため、それぞれが、少なくとも自動機械と、前記自動機械に関連するプログラミング可能な周辺機器とを含む、複数のセルから構成されるF Aシステムを制御するための制御方法において、前記複数のセルの少なくとも1のセルを構成する機器をそれぞれ複数のプロセッサにより制御するため、前記セル全体としての作業仕様を同一言語で記述してセル制御プログラムを作成し、当該作成したセル制御プログラムから、前記自動機械と周辺機器の作業順序の制御に関するプログラムと、前記自動機械の動作制御に関するプログラムと、前記プログラミング可能な周辺機器の制御に関するプログラムとに分離変換し、前記分離変換した各プログラムに基づいて、当該セルを構成する前記自動機械と前記プログラミング可能な周辺機器の動作をそれぞれ前記複数のプロセッサにより制御するF Aシステムの制御方法であって、当該プログラムの分離変換の際に、さらに、少な

くとも前記自動機械あるいは前記プログラミング可能な周辺機器の動作を制御するそれぞれの前記複数のプロセッサにおける処理負荷の配分を行うFAシステムの制御方法が提案されている。

【0012】さらに、本発明によれば、やはり上記の本発明の目的を達成するため、それぞれが、少なくとも自動機械と、前記自動機械に関連するプログラミング可能な周辺機器とを含む、複数のセルから構成されるFAシステムにおいて、前記複数のセルの少なくとも1のセルを構成する機器をそれぞれ複数のプロセッサにより制御するための制御プログラムを生成するためのFAシステムの制御プログラム生成方法であって、前記セル全体としての作業仕様を同一言語で記述してセル制御プログラムを作成し、当該作成したセル制御プログラムから、前記自動機械と周辺機器の作業順序の制御に関するプログラムと、前記自動機械の動作制御に関するプログラムと、前記プログラミング可能な周辺機器の制御に関するプログラムとに自動的に分離変換し、そして、少なくとも前記自動機械あるいは前記プログラミング可能な周辺機器の動作を制御するそれぞれの前記複数のプロセッサにおける処理負荷の配分を自動的行うFAシステムの制御プログラム生成方法が提案されている。

【0013】加えて、本発明によれば、上記の目的は、ロボットを含む自動機械と視覚センサなどのプログラミング可能な周辺機器とを含む複数機器の単位（セル）から構成されるFAシステムの制御方法において、前記セル全体としての作業仕様を直接記述し、前記複数機器の作業順序の制御に関する情報と、前記複数機器の入出力の制御に関する情報と、前記自動機械の動作制御に関する情報と、前記プログラミング可能な周辺機器の制御に関する情報と、前記複数機器間の同期をとるための動作タイミングに関する情報とを有するセル制御プログラムから、機能別モジュール分離変換手段によって、前記複数機器の作業順序に関する情報と、前記入出力の制御に関する情報を分離及び抽出して、前記複数機器の作業順序の制御と前記入出力の制御を行う処理が記述されたシーケンス制御プログラムモジュールを生成し、また、前記自動機械の動作制御と前記動作タイミングに関する情報を分離及び抽出して、前記自動機械の位置決め及び軌道の制御を行う処理が記述された自動機械制御プログラムモジュールを生成し、また、前記プログラミング可能な周辺機器の制御と前記動作タイミングに関する情報を分離及び抽出して、前記プログラミング可能な周辺機器の制御を行う処理が記述された周辺機器制御プログラムモジュールを生成し、さらにこのシーケンス制御プログラムモジュールと自動機械制御プログラムモジュールと周辺機器制御プログラムモジュールとを、それぞれシーケンス制御プロセッサ別モジュール分割手段と自動機械制御プロセッサ別モジュール分割手段と周辺機器制御プロセッサ別モジュール分割手段によって、前記セルを制

御するプロセッサの構成と性能に基づいて、各プロセッサの処理負荷の配分が最適となるようなプログラムモジュールに分割し、さらにこれらの分割したプログラムモジュールを各プロセッサが実行可能な形式のプログラムに変換して、シーケンス制御プロセッサ用プログラム群と自動制御プロセッサ用プログラム群と周辺装置制御プロセッサ用プログラム群とを生成し、これらの各プロセッサ用プログラム群を、それぞれを解釈実行するプロセッサ群に対して出力するFAシステムの制御方法によって達成される。

【0014】すなわち、本発明では、FAシステムを構成する、一つのまとまった複数機器からなる単位要素（以下、セルとする）を構築する際に、セルに含まれるロボットなどの自動機械と視覚センサなどのプログラミング可能な周辺機器とその他の機器を制御するプログラムをセル制御プログラムとして一本化して記述し、さらに、このセル制御プログラムから、FAシステムの制御装置のプロセッサの構成と性能に応じて、各プロセッサの処理負荷の配分が最適となるように、各プロセッサ用のプログラムを自動的に生成することで、従来のように制御装置の構成に応じて複数のプログラムを作成したり、幾つものプログラミング言語を習得したりする必要もなく、その結果として、FAシステムの制御プログラムの開発効率を向上することができるFAシステムの制御方法を提供することが可能となる。

【0015】

【発明の実施の形態】以下、添付の図面を参照して、本発明の実施の形態について詳細に説明する。図2は、本発明に係るFAシステムの制御装置における処理プロセスを模式的に示すブロック図である。一般に、FAシステムは、自動機器であるロボットを含む複数の機器から構成され、これら複数の機器は、それぞれ、複数の機器の単位としての「セル」と呼ばれている。この図2の場合には、制御対象であるFAシステムの機器であるセル116は、例えば、2体のロボット116a、116bと、視覚センサ116c、116dと、そして、その他の周辺機器116e、116fなどから構成されている。

【0016】かかるFAシステムのセル116の作業を制御するプログラミングを作成する場合、まず、このセル116全体としての作業仕様を記述したセル制御プログラム102を、コンピュータなどにより構成される、いわゆる、セル制御プログラム編集手段101によって作成する。なお、このセル制御プログラム編集手段101は、例えば、セル116の制御装置に接続された端末（図3に示す）の画面上において、セル116の各機器の動作シーケンスをベトリネットの形式で記述することができるグラフィカル・ユーザ・インタフェースを有するものである。また、セル制御プログラム102のベトリネット表現については、後に具体例を用いて説明す

る。

【0017】セル制御プログラム102には、(1)ロボット116a、116bの動作制御に関する情報、

(2)視覚センサ116c、116dの画像処理に関する情報、(3)その他の周辺装置に接続されるI/Oの制御に関する情報、(4)これら各機器の動作順序と各機器間の動作タイミングの同期に関する情報が、セル116全体の作業仕様として、一種類のプログラミング言語(以下、セル制御言語とする)によって記述されている。このようにセル116の作業仕様を一つのプログラムとして記述することで、セル116全体としての動作シーケンスを明示的に表現することができる。なお、セル制御言語の仕様は、前記の従来技術、特開平7-72920号公報に記載されているものに拡張を加えたものであり、シーケンス表現にペトリネットを応用しているところなど、基本的な部分は共通の手法を用いている。

【0018】このセル制御プログラム102は、まず、機能別モジュール分離変換手段103によって、(a)シーケンス制御プログラムモジュール104と、(b)ロボット制御プログラムモジュール105と、そして、(c)画像処理プログラムモジュール106の3個のモジュールに分離及び変換される。ここで、上記のシーケンス制御プログラムモジュール104とは、セル116全体としてのシーケンス制御と、各種周辺機器116e、116fなどのI/Oの制御に関するプログラムモジュールであり、ロボット制御プログラムモジュール105とは、ロボット116a、116bの動作制御に関するモジュールであり、そして、画像処理プログラムモジュール106とは、視覚センサ116c、116dの画像処理に関するプログラムである。なお、以下の説明では、これらのシーケンス制御、ロボット制御、画像処理などのように制御機能別に分類されたプログラムモジュールのことを、場合により、まとめて機能別プログラムモジュールと呼ぶ。

【0019】これら機能別プログラムモジュールは、それぞれが処理される制御用プロセッサ(プログラブルコントローラ、ロボットコントローラなどの独立したコントローラも、それぞれ広義の制御用プロセッサと考えるものとする)の構成に応じたプログラムモジュールに変換される。すなわち、本発明によれば、セル116の制御装置が複数の制御用プロセッサから構成されることを想定して、これらの各プロセッサの処理負荷の配分が最適になるように、機能別プログラムモジュールをプロセッサ別に分割することで、最終的に各プロセッサで実行されるプロセッサ別プログラムを生成する。具体的には、シーケンス制御プログラムモジュール104から、シーケンス制御プロセッサ別モジュール分割手段107によって、シーケンス制御プロセッサ別プログラム群110を生成する。また、ロボット制御プログラムモジュール105から、ロボット制御プロセッサ別モジュール

分割手段108によって、ロボット制御プロセッサ別プログラム群111を生成する。また、画像処理プログラムモジュール106から、画像処理プロセッサ別モジュール分割手段109によって、画像処理プロセッサ別プログラム群112を生成する。

【0020】そして、これらシーケンス制御プロセッサ別プログラム群110、ロボット制御プロセッサ別プログラム群111、画像処理プロセッサ別プログラム群112は、これら各プロセッサ用プログラム群110、111、112を解釈実行するプロセッサに対して出力される。すなわち、シーケンス制御プロセッサ用プログラム群110はシーケンス制御プロセッサ群113に対して、ロボット制御プロセッサ用プログラム114はロボット制御プロセッサ群114に対して、画像処理プロセッサ用プログラム群112は画像処理プロセッサ群115に対して出力され、それぞれのプロセッサがこれらのプログラムを解釈実行することで、セル116は制御される構成となっている。

【0021】図3は、上記図2に示したFAシステムの制御装置の構成要件のうち、セル制御プログラム編集手段101における端末の画面表示の一例を示したものである。図示のように、このセル制御プログラム編集手段101は、セル制御プログラム102における動作シーケンスを、いわゆる、ペトリネットの形式で編集するためのグラフィカル・ユーザ・インタフェースを有している。このセル制御プログラム102の記述言語であるセル制御言語は、セル116の動作シーケンスの記述に対して、ペトリネットの図的表現形式を応用した一種のグラフィック言語である。この図3の例では、ペトリネットのプレース(図中、「○」で表記)をセル116を構成する機器の動作状態、トランジション(図中、「-」で表記)を次の動作状態に移移するときの遷移条件、トークン(図中、表記せず)の置かれたプレースを現在の活性化された状態(以下、活性化状態とする)と定義し、各プレースには高々一個のトークンしか存在しないセーフペトリネットを考えている。

【0022】なお、図3のネット編集ウィンドウ201では、マウス等のポインティングデバイスを用いて、画面上でグラフィックイメージのまま、ペトリネットを編集することができる。コマンド編集ウィンドウ202は、ネット編集ウィンドウ201の中でマウスのカーソルが指示している編集中の状態S106における機器の動作(ロボットの動作命令、I/OのON/OFFなど)やその他の処理を定義するためのコマンドを編集するためのウィンドウである。状態終了条件編集ウィンドウ203は、編集中のS106の正常終了の条件(OK条件)及び異常終了の条件(NG条件)を編集するためのウィンドウである。それぞれのウィンドウには、マウスやファンクションキーによるメニュー選択形式を利用した簡便な入力環境が用意されている。

【0023】図4には、上記したセル制御プログラム編集手段101によって作成されたペトリネットの一例を示す。なお、この図4のペトリネットは、一例として、以下の図5及び図6に示す電子部品の挿入作業を表したものである。

【0024】これら図5及び図6に示す作業は、プリント基板406にIC等の電子部品405を挿入するもので、まず、コンベアなどで搬送されてきたプリント基板406を位置決めし、パーツフィードなどによって供給される部品405を部品挿入ロボット401の先端に設けられたハンド402により把持し、所定の位置へ挿入する。さらに、挿入の終了した部品405のリードを、プリント基板406の下からアクセスするクリンチロボット407の先端のクリンチハンド408によって曲げ、部品405がプリント基板406から脱落するのを防止する。また、それぞれのロボット401、407のプリント基板406に対する位置補正のために、視覚センサ404、409によって位置ずれ計測を行う。図5は、このような部品挿入直前における各機器の状態を示し、図6は部品挿入直後の状態を示している。

【0025】上記図4においてペトリネットで表されたセル全体の動作シーケンスは、大きく分けると、部品挿入ロボット401（以下、「ROB1」と略記）のモジュール304、クリンチロボット407（以下、「ROB2」と略記）のモジュール305、部品挿入用の視覚センサ404（以下、「CAM1」と略記）のモジュール306、クリンチ用の視覚センサ409（以下、「CAM2」と略記）のモジュール307の4個のモジュールから構成される。各モジュールは、あるまとまった機器群で、その機器群同士で互いに並行動作可能なもの（以下、このような機器群のことを「ユニット」とする）毎にまとめられたシーケンスであり、これらのモジュールは同期プレース303（図中、二重丸で表記）で接続されることで、互に関連付けられる。

【0026】次に、上記図4のプレース301は、ユニットの動作状態を示しており、これを状態プレースと呼ぶ。例えば、状態プレース301に付記された「S106」は、動作状態の整理番号を示す。この状態プレース301における動作の具体的な内容は、ロボット命令やI/O制御命令などのコマンド列として定義する。また、トランジション302は、ある状態から次の状態に遷移するための条件を示している。トランジション302の遷移条件は、ペトリネットの接続関係と各状態の終了条件によって決まる。なお、このトランジション302に付記されている「OK1」は、このトランジション302への入力となる状態の正常終了を意味し、この正常終了の判定条件は、状態終了条件の式として定義される。同期プレース303は、モジュール間の動作タイミングの同期を定義するものであり、例えば、「REQ6」は、状態S105が正常終了し、かつ、状態S20

4が正常終了したとき、次の状態S106への遷移条件が満たされる、という二つの状態終了の同期をとる。なお、図4の状態プレース301の横には、その状態における動作の内容を示すコメントを付記している。

【0027】上記図4に示す動作シーケンスの概略を以下に説明する。まず、ROB1が、ハンド402で把持した部品405を、位置決めされたプリント基板406の上の所定の位置（挿入点）まで移動する（S100）と、CAM1がROB1の目標位置（プリント基板406との相対的な位置）からの位置ずれを計測し（S151）、その位置ずれ量に基づいてROB1は位置補正の動作を行う（S102）。さらに、ROB1は、そのリストを所定の高さまで下降させた後、ハンド402を開いて部品103を解放する（S103）。この間にROB2は、プリント基板406の直下の所定の位置（クリンチ点）まで移動（S200）し、CAM2によって位置ずれ計測（S251）した結果に基づいてROB2は位置補正を行い（S202）、さらに、そのリストを所定の高さまで上昇させる（S203）。ROB1のハンド402に設けられたセンサ（図示せず）によって部品405の解放が検知されると、ROB1の手先に設けられたセンタプッシャ403が下降することで部品405を押し下げ、プリント基板406の所定の穴に部品405を深く挿入する（S104）。ROB1の手先に設けられたセンサ（図示せず）によりセンタプッシャ403が所定の高さまで下降したことが検知されると、ROB2のクリンチハンド408が閉じて、部品405のリードが折り曲げられる（S204）。クリンチハンド408に設けられたセンサ（図示せず）により、クリンチが完了したことが検知されると、クリンチハンド408が開かれ（S205）、ROB1のセンタプッシャ403が上昇する（S106）。その後、ROB1は挿入準備点まで移動し（S107）、ROB2はリストを下降し（S206）、クリンチ準備点まで移動する（S207）。

【0028】次に、図7と図8には、上記図3にも説明したセル制御プログラム編集手段101によって、図4のペトリネットを編集し、それをコード化したもの、すなわち、図2のセル制御プログラム102の一例を示す。図からも明らかなように、セル制御プログラム102は、2種類のブロックから構成されており、一つは図7に示すcellブロック（“cell”文に続くブロック）であり、もう一つは、図8に示すdefブロック（“def”文に続くブロック）である。

【0029】このセル制御プログラム102について、さらに、その詳細な内容を説明すると、図7のcellブロックには、ペトリネットの接続関係と各状態における機器の動作が定義されている。cellラベル601には、状態の整理番号とその状態における動作主体であるユニットの種別とその整理番号が定義される。例え

ば、状態S106のユニットはROB1であり、これは部品挿入ロボット401とそれに付随するハンド402等の周辺機器から構成されるユニットを示す。遷移条件式602は、ペトリネットの接続関係を定義する式であり、その状態の入カトランジションに入力として接続される状態の整理番号とその状態の終了条件を定義する。状態S106の入カトランジションは、状態S105の正常終了OK1と同期プレースREQ6のOKとに接続されている。コマンド列603は、その状態におけるユニットを構成する機器の動作とその他の処理を定義するコマンド列である。状態S106における処理は、RESET Y101（出力Y101のOFF、すなわちセンタプツシャ403の上昇）と、SET TD1210.（オンディレイタイマTD12のセット）である。

【0030】また、図8のdefブロックには、各状態の終了条件が定義される。defラベル701には状態の整理番号が定義され、定義式702には状態の正常終了（OK）と異常終了（NG）の条件式が定義される。状態S106の正常終了条件（OK1）は、X101=OFF（入力X101がOFF、すなわち部品405の接触検出のセンサがOFF）かつTD12=OFF（オンディレイタイマTD12がOFF、すなわちタイムアップしていない）である。状態S106の異常終了条件（NG1）は、X101=ONかつTD102=ON（オンディレイタイマTD102がタイムアップしても、センサがONのまま）である。なお、図中の「＝」は、条件式であることを示す。

【0031】このように、上記図4にも示すように、セル制御プログラム102の記述にペトリネットの表現を応用することで、各機器の動作シーケンスが構造的かつ明示的に記述することができる。さらに、図7、図8のように、セル内の機器の制御プログラムをセル制御プログラム102として一本化し、しかも、各機器の動作シーケンスの明示性を持たせることで、FAシステムのソフトウェアの開発者にとって、セル内の機器の制御装置の構成やプログラム言語の違いを意識せずに、セル全体としての作業仕様を直接記述する形で、セル制御プログラム102を容易に作成することが可能になる。すなわち、セル制御プログラム102の開発者は、従来のようにいくつかの異なったプログラム言語を習得する必要がなくなり、その結果、プログラムの開発効率が向上し、ソフトウェアとしての保守性も向上する。

【0032】そして、本発明によれば、上記図7、図8のようなセル制御プログラム102から、機能別モジュール分離変換手段103によって、セルを構成する機器の動作順序の制御とこれらの機器に接続されたI/Oの制御を行う処理が記述されたシーケンス制御プログラムモジュール104と、ロボット116a、116bの位置決め及び軌道の制御を行う処理が記述されたロボット制御プログラムモジュール105と、視覚センサ1

16c、116dの画像に関する処理を記述した画像処理プログラムモジュール106を生成する。

【0033】図9には、上記機能別モジュール分離変換手段103の処理プロセスを示す。なお、セル制御プログラム102のcellブロックには、上記の図7及び図8に示したように、セル制御プログラム102のcellブロックには、ペトリネットにおける各プレースの接続関係、すなわちネット構造と、各状態における機器の動作が定義され、また、defブロックには、各状態の終了条件が定義される。

【0034】図の機能別モジュール分離変換手段103の機能別データ抽出手段103aは、まず、cellブロックの内容からネット構造103bを、defブロックの内容から状態終了条件103cを抽出する。さらに、cellブロックからI/O制御命令列103dを抜き出す。そして、シーケンス制御プログラム生成手段103iは、上記抽出したネット構造103b、状態終了条件103c、そして、I/O制御命令列103dから、シーケンス制御プログラムモジュール104を生成する。

【0035】また、機能別データ抽出手段103aは、cellブロックの内容から、ロボット制御命令列103eとロボット動作タイミング103fに関する情報を抽出する。そして、ロボット制御プログラム生成手段103jは、これら抽出した情報に基づいて、ロボット制御プログラムモジュール105を生成する。

【0036】同様に、機能別データ抽出手段103aは、cellブロックの内容から画像処理命令列103gと画像処理タイミング103hに関する情報を抽出し、さらに、画像処理プログラム生成手段103kは、これらの情報に基づいて画像処理プログラムモジュール106を生成する。

【0037】図10、図11、及び、図12には、上記図7、図8に例示したセル制御プログラム102から、上記図9に示した機能別モジュール分離変換手段103によって生成された、シーケンス制御プログラムモジュール104の一例を示す。なお、本例におけるシーケンス制御用のプログラミング言語は、IF～THEN～形式のルール型の記述を基本とする。すなわち、各ルールには、各状態における機器の動作の実行ステータス、センサなどの外部からの入力信号、内部変数の値などについての評価式を条件とし、ここで所定の条件が満たされれば、その状態における機器の動作を終了し、次の段階の状態における機器の動作を起動する、という処理が記述される。例えば、図11中に示したルール1001は、状態S203が正常終了し、かつ同期プレースREQ5が正常終了すれば、状態S204の実行ステータスを実行中にし（EXEC S204）、出力Y200とオンディレイタイマTD20をセットすることを表す。

【0038】なお、シーケンス制御用のプログラミング

言語としては、本例のようなルール型のプログラミング言語の他にも、例えばラダー図などのプログラミング言語を用いることも可能であり、その場合、機能別モジュール分離変換手段103は、シーケンス制御プログラムモジュール104としてラダープログラムを生成することとなることは当業者にとっては当然であろう。

【0039】図13には、上記図7、図8のセル制御プログラム102から、上記機能別モジュール分離変換手段103によって生成された、ロボット制御プログラムモジュール105の一例を示す。この図13の中に示したブロック1203は、状態S203(STT203)において、ROB2のリストを10mmだけ上昇させる(MOVE L, *+(0., 0., 10.))ことを表す。ここで、STTn(nは状態Snのnと同じ)は、先に示したシーケンス制御プログラムモジュール104と共通する状態の整理番号(以下、状態NO.とする)を示す状態ラベルであり、この状態NO.を各プログラムの中で共有することによって、セル全体の状態遷移を制御することができる。

【0040】図14には、上記図7、図8のセル制御プログラム102から、上記機能別モジュール分離変換手段103によって生成された画像処理プログラムモジュール106の一例を示す。ここで生成される画像処理プログラムモジュール106は、ロボット制御プログラムモジュール105と同様の構成を持っており、STTn文に続けて各種の画像処理命令列が記述されている。

【0041】さらに、図15～図24には、上記機能別モジュール分離変換手段103の処理の詳細を説明するフローチャートを示す。まず、図15は、上記機能別モジュール分離変換手段103における機能別モジュール分離変換処理の概略を示す。この機能別モジュール分離変換処理では、処理がスタートすると、まず、セル制御プログラムを解析して機能別データの抽出を行い(処理1100a)、ここで得られたデータに基づいて、機能別モジュールを生成する。すなわち、シーケンス制御プログラム生成手段103iによってシーケンス制御プログラムモジュール104を生成し(処理1100b)、ロボット制御プログラム生成手段103jによってロボット制御プログラムモジュール105を生成し(処理1100c)、画像処理プログラム生成手段103kによって画像処理プログラムモジュール106を生成する(処理1100d)。

【0042】図16は、上記図15に示した機能別モジュール分離変換処理の機能別データ抽出処理1100aの詳細を示す。この機能別データ抽出処理1100aでは、処理がスタートすると、セル制御プログラムのcellブロック、defブロックの順にコード解析を行い、それぞれ必要なデータ抽出を行う。まず、cellブロックの解析では、各ロボット毎の動作タイミング103fとロボット制御命令列103eを格納するための

データ領域の確保(処理1101a)、各視覚センサ毎の画像処理タイミング103hと画像処理命令列103gを格納するためのデータ領域の確保(処理1101b)、I/O制御命令列103dを格納するためのデータ領域の確保(処理1101c)を行う。次に、セル制御プログラムを、ペトリネットの1プレースに相当するブロック単位で、先頭から順に処理していき、プレースの接続関係、すなわち、ネット構造103bをコード解析によって抽出し(処理1101d)、さらに、各プレースにおいて定義されるコマンド列を解析する(処理1101e)。これらの解析は、次のプレースのブロックがなくなるまで繰り返す(処理1101f)。defブロックの解析では、1プレース毎に状態の終了条件103cをコード解析によって抽出し(処理1101g)、次のプレースのブロックがなくなるまで、これを繰り返して(処理1101h)、終了する。

【0043】次に、図17及び図18は、1プレース毎のネット構造解析処理101dの詳細を示す。この1プレース毎のネット構造解析処理101dでは、まず、1プレース毎のネット構造データ領域を確保し(処理1102a)、次に、cellラベル601と遷移条件式602の解析を行う。図17のcellラベルの解析では、まず、そのcellラベルが状態プレースのラベルか、あるいは、同期プレースのラベルかを調べ(処理1102b)、その結果、状態プレースであれば、その状態NO.を抽出し、ネット構造データの一つとして格納する(処理1102c)。さらに、cellラベルからユニット種別とユニットNO.を抽出し、ネット構造データとして格納する(処理1102d、1102e)。ここで、ユニット種別がロボットであれば(処理1102f)、ロボット動作タイミング103fのデータとして先に抽出した状態NO.を格納する(処理1102g)。また、ユニット種別が視覚センサであれば(処理1102h)、画像処理タイミング103hのデータとして先の状態NO.を格納する(処理1102i)。一方、cellラベルが同期プレースのラベルであれば(処理1102b)、その同期プレースの整理番号(以下、同期NO.)を抽出し、ネット構造データとして格納する(処理1102j)。

【0044】一方、図18の遷移条件式の解析では、まず、遷移条件式602(上記図7の例では、S105:OK1 REQ6:OK)から状態NO.(S105)と終了条件(OK1)を抽出し、これらをネット構造データとして格納する(処理1103a、1103b)。ここで、同期条件があれば(処理1103c)、同期NO.(REQ6)を抽出し、ネット構造データとして格納する(処理1103d)。さらに、別の遷移条件があれば、その遷移条件について同様の解析処理を行い、遷移条件がなければ解析処理を完了する(処理1103e)。

【0045】以上のように、上記図17と図18に示したネット構造解析処理により、与えられたセル制御プログラムと等価なペトリネットの各プレースの接続関係、すなわち、ネット構造103bの情報が得られる。

【0046】次に、図19は、上記1プレース毎のコマンド列解析101eの詳細を示す。ここでは、cellラベル601（図7を参照）と遷移条件式602に続くコマンド列603の解析を行う。まず、コマンド列603のブロックから1コマンドを抽出し（処理1104a）、これがI/O制御命令であれば（処理1104b）、I/O制御命令列データとして格納する（処理1104c）。また、抽出したコマンドがロボット制御命令であれば（処理1104d）、これをロボット制御命令列データとして格納する（処理1104e）。抽出したコマンドが画像処理命令であれば（処理1104f）、画像処理命令列データとして格納する（処理1104g）。なお、次のコマンドがあれば、処理1104a以降を繰り返し、コマンドがなければ、解析処理を完了する（処理1104h）。

【0047】図20は、上記1プレース毎の終了条件解析101gの詳細を示す。ここでは、セル制御プログラムのdefブロックに定義された各プレース（状態）の終了条件の解析を行う。まず、1プレース毎の終了条件データを格納する領域を確保する（処理1105a）。さらに、defラベル701から状態NO.（図8の例では、S106）を抽出する（処理1105b）。次に、defラベル701に続くOK条件の定義式（図8の例では、X101=OFF & TD12=OFF）を抽出し、これを終了条件データとして格納する（処理1105c）。次のOK条件がなくなるまで、この処理を繰り返す（処理1105d）。さらに、NG条件の定義式（図8の例では、X101=ON & TD12=ON）を抽出し、これを終了条件データとして格納する（処理1105e）。次のNG条件がなくなるまで、この処理を繰り返す（処理1105f）。以上の処理により、各状態の終了条件を得ることができる。

【0048】図21及び図22は、上記シーケンス制御プログラム生成処理100bの詳細を示す。ここでは、先の機能別データ抽出処理100aによって得られたネット構造103b、状態終了条件103c、I/O制御命令列103dのデータに基づいて、各ユニット毎にシーケンス制御プログラムを生成する。まず、各ユニット毎のプログラムバッファ領域を確保する（処理1106a）。次に、先に生成したネット構造データを1プレースずつ順に調べ、プレース毎にシーケンス制御プログラムを生成していく。

【0049】そこで、まず、1プレース毎のネット構造データからプレース種別を獲得し（処理1106b）、そのプレースが状態プレースであれば（処理1106c）、その状態NO.（整理番号をiとおく。以下、

「=i」と略記）とユニットNO.を獲得する（処理1106d）。そのプレースが状態プレースでなければ、ネット構造データから同期NO.（=i）を獲得する（処理1106e）。さらに、ネット構造データと終了条件データから、現在着目しているプレースに接続する、一つ前の状態プレースの状態NO.（=j）と終了条件の定義式を獲得する（処理1106f）。この一つ前の状態の終了条件が、“CompleteNormally”であれば（処理1106g）、“IF Sj=CompleteNormally”をプログラムバッファに出力し（処理1106h）、“AnyErrorOccur”であれば（処理1106i）、“IF Sj=AnyErrorOccur”をバッファに出力する（処理1106j）。それ以外の定義式の場合、“IF Sj=Executing & (定義式)”をバッファに出力する（処理1106k）。

【0050】ここで、現プレースが状態プレースであれば（処理1106l）、現プレースに接続する同期プレースがあるかどうか調べ、同期プレースがあれば（処理1107a）、その同期NO.（=k）をネット構造データから獲得し（処理1107b）、“& REQk=CompleteNormally”をバッファに出力する（処理1107c）。以上の処理で、1プレースあたりのシーケンス制御プログラム、すなわち、ルール1001（図11を参照）の条件部が生成される。

【0051】次に、ルール1001の実行部として、まず、“→ TERM Sj”をバッファに出力する（処理1107d）。ここで、現プレースに接続する同期プレースがあれば（処理1107e）、“TERM REQk”をバッファに出力する（処理1107f）。さらに、“EXEC Si”をバッファに出力し（処理1107g）、現プレース（Si）のI/O制御命令列データの内容をバッファに出力する（処理1107h）。また、現プレースが同期プレースであれば（処理1106l）、“→ TERM Sj”と“EXEC REQi”をバッファに出力する（処理1107i、1107j）。以上の処理で、ルール1001の実行部が生成される。この後、さらに、ネット構造データを調べ、次のプレースが見つければ（処理1107k）、処理1106b以降を繰り返す。次のプレースがなければ、各ユニット毎のプログラムバッファを結合し、シーケンス制御プログラムモジュール104（図9を参照）を生成する（処理1107l）。

【0052】図23は、上記ロボット制御プログラム生成処理100cの詳細を示す。まず、各ユニット毎のプログラムバッファ領域を確保する（処理1108a）。次に、先の機能別データ抽出処理100aで得られたユニット毎のロボット動作タイミング103fのデータから、状態NO.（=i）を一つずつ獲得し（処理1108b）、状態ラベル“STTi”をプログラムバッファ

に出力する(処理1108c)。さらに、上記の機能別データ抽出処理100aで得られたロボット制御命令列103eのデータがあれば(処理1108d)、そのロボット制御命令列データの内容をバッファに出力する(処理1108e)。ロボット制御命令列データがなければ、“NOP”をバッファに出力する(処理1108f)。以上の処理で1状態分のロボット制御プログラムが生成される。ここで、ロボット動作タイミングデータを調べ、次に続く状態が見つければ(処理1108g)、処理1108b以降を繰り返す。次の状態がなければ、これで1ユニット分のロボット制御プログラムが生成されたことになる。さらに、次のユニットのロボット動作タイミングデータがあるかどうか調べ、次のユニットのデータがあれば(処理1108h)、処理1108b以降を繰り返す。次のユニットのデータがなければ、各ユニット毎のプログラムバッファを結合し、ロボット制御プログラムモジュール105(図9を参照)を生成する(処理1108i)。

【0053】図24は、画像処理プログラム生成処理100dの詳細を示す。なお、図からも明らかなように、ここでは、上記図23と同様の処理により、図15の機能別データ抽出処理100aで得られた画像処理タイミング103h(図9を参照)のデータと、画像処理命令列103gのデータから、画像処理プログラムモジュール106を生成する。

【0054】次に、図25は、セル制御プログラム102(図9を参照)からシーケンス制御プログラム104と、ロボット制御プログラムモジュール105とを生成する手順を、上記図7、図8に示したセル制御プログラムの一部を例にして、具体的に示すものである。

【0055】この図25の符号1401、1402は、それぞれ、上記セル制御プログラム102のcellブロック、defブロックの一部を示す。このcellブロック1401において、先に述べたcellラベル601(図7を参照)と状態遷移式602を合わせたものが、この図中のベトリネットの状態接続関係1401a、1401cである。この状態接続関係1401aは、状態S106への入力トランジションが一つであり、そこに状態S105の正常終了OK1と同期プレースREQ6のOKが入力として接続されていることを示す。同様に、状態接続関係1401cは、状態S107のトランジションは一つで、そこに状態S106のOK1が接続されていることを示す。

【0056】また、defブロック1402には、各状態の終了条件が定義されており、状態S105の正常終了OK1の状態終了条件1402aは、状態S105における動作がエラーなしで完了することを示す。同様に、状態S106のOK1の状態終了条件1402cは、入力X101がOFFであり、かつ、オンディレイタイマTD12がタイムアップしていないことを示す。

【0057】これら状態接続関係1401aと状態終了条件1402aから、シーケンス制御プログラム1403におけるルール条件部1403aと、実行部の状態遷移命令1403b(EXEC S106など)が生成される(処理1405、1406)。すなわち、ルール条件部1403aとして「もし、状態S105が正常終了であり、かつ同期プレースREQ6が満たされれば」が生成され、状態遷移命令1403bとして「状態S106を実行する」などが生成される。同様にして、状態接続関係1401cと状態終了条件1402cから、ルールの条件部1403dと、実行部の状態遷移命令1403eが生成される(処理1407、1408)。

【0058】また、このとき同時に、状態接続関係1401a、1401cから、状態NO。すなわちロボット動作タイミング103fが抜き出され、ロボット制御プログラム1404の中の状態ラベル1404a、1404cが生成される(処理1409、1410)。

【0059】さらに、cellブロック1401のコマンド列から、I/O制御命令列1401bが抜き出されて、シーケンス制御プログラム1403のルールの後件部のコマンド列1403cが生成される(処理1411)。さらに、cellブロック1401のコマンド列のうち、ロボット制御命令列1401dが抜き出され、ロボット制御プログラム1404のコマンド列1404b、1404dが生成される(処理1412、1413)。

【0060】図26は、セル制御プログラム102(図9を参照)から、シーケンス制御プログラム104と画像処理プログラムモジュール106を生成する手順を、やはり、図7、図8に示したセル制御プログラムの一部を例にして、具体的に示すものである。すなわち、上記の図25と同様に、セル制御プログラム102のcellブロック1501と、defブロック1502とから、シーケンス制御プログラム1503と、画像処理プログラム1504とを生成する手順を示している。

【0061】以上のようにして、機能別モジュール分離変換手段103によって生成される機能別プログラムモジュールは、共通のプログラム構造を持ち、その構造を図27に示す。すなわち、図27に示すように、そのプログラム構造において、機能別プログラムモジュール1601は、いくつかのユニットブロック1602によって構成され、さらに、このユニットブロック1602は、状態ブロック1603によって構成される。このユニットブロック1602は、ユニットラベル(UNITn、#ROBn、#CAMnなど、nはユニットの整理番号)に続くプログラムのブロックである。また、状態ブロック1603は、シーケンス制御プログラムではルール1001に相当し、ロボット制御プログラムと画像処理プログラムでは状態ラベル(STTn、nは状態NO。)に続くブロックに相当する。

【0062】例えば、図11に例示したシーケンス制御プログラムは、ROB1のユニットブロック1602であり、ルール1001は状態S203の状態ブロック1603に相当する。また、図14のロボット制御プログラムは、ROB1のユニットブロック1201と、ROB2のユニットブロック1202から構成されており、また、図中のブロック1203は、図27の状態S203の状態ブロック1603に相当する。また、図14の画像処理プログラムも、図13と同様、図27に示す、CAM1のユニットブロック1602と、CAM2のユニットブロック1602とから構成されている。

【0063】この図27に示した機能別プログラムモジュール1601のブロック構造は、この機能別プログラムモジュール1601を分割することで得られるプロセッサ別プログラムのコード実行手順と密接な関係がある。すなわち、プログラム実行時には、あるユニットブロック1602の中のうち、現在活性化されている状態ブロック1603が選出され、そのブロックに記述された命令列が順番に実行される。さらに、別のユニットブロック1602においても、同様の処理が行われる。ここで、処理中のユニットブロック1602を指すユニットラベルの番号（#UNITnのnなど）は、ユニットそのものの整理番号（以下、ユニットNO. とする）と同じものであり、このユニットNO. は、プログラムの実行時にプログラム制御を行うための情報（以下、プログラム制御用データとする）として管理される。また、機器の現在の状態、すなわち活性化状態における動作が記述された状態ブロック1603を指す状態ラベルの番号は、状態NO. として管理される。

【0064】次に、図28には、このプロセッサ別プログラムの実行手順を示す。同図に示す処理フローにおいて、まず、現在のユニットNO. をプログラム制御用データから読み込み（処理1701）、そのユニットNO. と同じ番号を持つユニットラベルへジャンプする（処理1702）。次に、活性化状態の状態NO. を読み込み（処理1703）、その状態ラベルへジャンプする（処理1704）。この状態ラベルに続く命令列を順番に実行し（処理1705、1706）、その実行が終了すると次のユニットブロック1602を処理するためにユニットNO. を更新し（処理1707）、再びフローの最初に戻る。このように、プログラムはユニットブロック1602を単位として実行されるので、機能別プログラムモジュール1601はこのユニットブロック1602を単位として、さらに分割することができる。また、先に述べたプロセッサ別プログラムは、機能別プログラムモジュール1601を、上記ユニットブロック1602を単位として分解することによって、生成される。

【0065】続いて、図1には、上記の機能別プログラムモジュール1601から、プロセッサ別プログラム群

1803を生成する、いわゆる、プロセッサ別モジュール分割手段1801の処理プロセスを示す。このプロセッサ別モジュール分割手段1801では、まず、機能別プログラムモジュール1601をユニットブロック分割手段1801aによって、個々のユニットブロック1602に分解して、ユニットブロック群1801bを生成する。

【0066】なお、ここで、実際に機器を制御するプロセッサ構成及びその性能に関するデータ（プロセッサ構成及び性能データ）1801c、すなわち、制御用プロセッサを何台用いて、それらをどのような形態で接続するか、ということに関する情報と、制御用プロセッサのプログラムコードの処理時間などに関する情報に基づいて、各ユニットブロック1602（図27を参照）の処理時間を推定し、それが制御上必要とされる処理時間（例えば、シーケンス制御におけるスキャンタイムやロボット制御におけるサンプリング周期など）の基準を満たしているかどうかを評価する。これを行うのがユニットブロック処理時間推定及び評価手段1801dであり、これにより各ユニットブロック1602の処理時間に関するデータが作成される。なお、プログラムコードの処理時間については、各命令コードの処理に要する時間（制御用プロセッサ毎に予め調べられているものとする）を実行される命令コードの分だけ加算することで求めることができる。例えば、シーケンス制御プロセッサにおけるスキャンタイム（1処理に要する時間）であれば、1スキャンで実行する分だけ命令コードの処理時間を加算することにより、スキャンタイムを推定することができる。

【0067】次に、ユニットブロック結合手段1801eは、各ユニットブロック1602の処理時間の推定データに基づいて、いくつかのユニットブロック1602を結合し、最終的に各プロセッサで実行されるプロセッサ別プログラム群1803を生成する。本実施例では、図28のようなプログラムコードの実行手順をとるため、各プロセッサにおいては、その制御上必要とされる処理時間の基準を満たす範囲内で、複数のユニットブロック1602を実行することが可能である。すなわち、いくつかのユニットブロック1602の推定処理時間を加算して、それが制御上の基準の範囲内であれば、それらのユニットブロック1602を結合し、それをプロセッサ別プログラム1802とすることができる。ただし、ユニットブロック1602を結合するとき、特定のプロセッサに処理負荷が集中しないように、プロセッサの性能に応じた最適な負荷配分がされていることが望ましい。そこで、ユニットブロック結合手段1801eは、このように処理負荷の配分が最適になるように、ユニットブロック1801eを結合して、プロセッサ別プログラム群1803を生成する。すなわち、個々のプロセッサの能力をオーバーしないよう、かつ、最大時間を

要するプロセッサの処理時間が最小になるように処理負荷を配分する。

【0068】図29には、上記のプロセッサ別モジュール分割手段1801によって、機能別プログラムモジュール1601を分割して、プロセッサ別プログラム群1802を生成した例を示す。ここでは、機能別プログラムモジュール1601に含まれる3個のユニットブロック1901、1902、1903から、2個のプロセッサ別プログラム1904、1905を生成した例を示している。

【0069】既述の図10、図11、図12に示したシーケンス制御プログラムを例に取って説明すると、これらのプログラムを実行するシーケンス制御プロセッサが1個であり、これらのプログラムの全てを実行した場合のスキヤンタイムが所定の時間以内であれば、これらのプログラムを一つのシーケンス制御プログラムとして結合する。これに対し、例えばシーケンス制御プロセッサが2個の場合、例えば、図10を一つのプログラム、図11と図12を結合して一つのプログラムとし、これら二つのプログラムをそれぞれのシーケンス制御プロセッサに対して出力する。

【0070】なお、上記図1に示したユニットブロック結合手段1801eにおいて生成されるプロセッサ別プログラム1802は、さらに、その出力先となるプロセッサで実行可能な形式のプログラムコードに変換する必要があることも想定される。このような場合、ユニットブロック結合手段1801eの後に、各プロセッサ用言語に変換するためのプロセッサ別のコード変換手段を予め設けることで対応することができる。

【0071】続いて、図30には、以上のようにしてセル制御プログラム102（図9を参照）から生成されたプロセッサ別プログラム群1803（図1を参照）、すなわち、シーケンス制御プロセッサ用プログラム群110、ロボット制御プロセッサ用プログラム群111、画像処理プロセッサ用プログラム群112（図2を参照）を解釈実行し、セルを制御するFAコントローラの一例を示す。なお、この図30に示されるコントローラは、シーケンス制御プロセッサ群113、ロボット制御プロセッサ群114、画像処理プロセッサ群115を共有バス2004によって結合した、いわゆる、マルチプロセッサ型コントローラ2001を構成している。また、図中の符号2014はセルを示しており、このセルは、周辺機器（I/O）2011、ロボット2012、視覚センサ2013を含んでいる。

【0072】この図の構成において、シーケンス制御プロセッサ2005は、共有バス2004とのインタフェースであるバスドライバ2005d、さらに、シーケンス制御プロセッサ用プログラムを解釈実行する演算手段2005aなどを含んで構成されており、また、I/Oインタフェース2008を介して、その制御対象である

周辺機器（I/O）2011に接続されている。

【0073】ロボット制御プロセッサ2006は、バスドライバ2006f、ロボットの手先の位置姿勢を教示するための教示手段2006b、ロボット制御プロセッサ用プログラムを解釈実行する演算手段2006aなどから構成され、モータ駆動回路2009を介して、その制御対象であるロボット2012に接続される。

【0074】そして、画像処理プロセッサ2007は、バスドライバ2007f、基準画像を教示するための教示手段2007b、画像処理プロセッサ用プログラムを解釈実行する演算手段2007aなどから構成され、画像インタフェース2010を介して、工業用テレビカメラなどで構成される視覚センサ2013に接続されている。

【0075】さらに、上記のシーケンス制御プロセッサ群113、ロボット制御プロセッサ群114、画像処理プロセッサ群115で実行されるプログラムには、機器の動作タイミングに関する情報が組み込まれており、各プログラムを実行する際に、このような動作タイミングの同期を取るためには、各プロセッサの間で必要なデータを共有する必要がある。そこで、各プロセッサがアクセスできる共有バス2004には、共有データ記憶手段2003が接続されている。

【0076】図31には、上記共有データ記憶手段2003に格納される共有データ2101の構成を示す。先に述べたように、セル制御プログラム102（図9を参照）では、各ユニットの動作シーケンスを状態遷移の考え方に基づいて記述しているので、それから生成された各プログラムを実行する際にも、活性化された状態NO.を各プロセッサの間で共有することによって、ユニットの動作タイミングの同期をとることができる。そこで、共有データ2101には、ユニット別状態データ2102として、活性化された状態NO. 2101aと、その実行ステータス2102b（活性化状態に関するより詳細な情報）が格納されている。各プロセッサは、それぞれのプログラムを実行中に、このユニット別状態データ2102にアクセスすることで、各プログラム中に記述された動作タイミングの同期をとることができる。また、共有データ2101には、ユニット別状態データ2102の他に、各プロセッサ間で共有すべき各種データを、共有変数2103として格納することができる。

【0077】なお、上記の図30に示したマルチプロセッサ型コントローラ2001は、通信手段2002によって、例えばパーソナルコンピュータなどのホストコンピュータ2015と接続される。このホストコンピュータ2015には、セル制御プログラム編集手段101（図2を参照）や機能別モジュール分離変換手段103、プロセッサ別モジュール分割手段1801のソフトウェアを搭載することが可能であり、ここで作成されたプログラムは、上記の通信手段2002によって各プロ

セッサに送信することができる。

【0078】また、図32は、上記図30に示したマルチプロセッサ型コントローラ2001とは異なり、従来の一般的な構成を持つFAシステムの制御装置の構成を示すものである。すなわち、セル2207を制御するために、プログラマブルコントローラ2201、ロボットコントローラ2202、画像処理装置2203をネットワーク2208で接続した例である。このように幾つかのコントローラを組み合わせた構成の場合でも、図31に示したような共有データ2101を、ネットワーク2208を通じて、それぞれのコントローラ同士で交換し、共有することで、ユニット間の動作タイミングの同期をとることができる。よって、本発明の制御方法は、従来型の構成を持つ制御装置にも適用することが可能である。

【0079】また、図33には、さらに別の構成を持つFAシステムの制御装置を示す。これは複数のマルチプロセッサ型コントローラ2301、2302をネットワーク2306で接続したもので、この場合でも、各コントローラが、ネットワーク2306を経由して、図31に示したような共有データ2101を交換することで、ユニット間の同期を取ることができ、よって、図33のようなセル集合2305、つまり、より規模の大きいFAシステムに対しても、本発明は適用可能であることが理解される。

【0080】さらに、上記の図30に示したようなマルチプロセッサ型コントローラ2001と、上記の図32に示したような従来型のコントローラが混在した制御システムの場合でも、上記の図33と同様にして、本発明は適用可能である。なお、上記した実施の形態の例では、自動機械としてロボットを含んだセルを対象としたが、ロボット以外の自動機械、例えばNC工作機械などを含んだセルに対しても、同様にして本発明を適用することは可能である。

【0081】また、上記の例ではプログラミング可能な周辺機器として視覚センサを含んだセルを対象としたが、視覚センサ以外のプログラミング可能な周辺機器を含んだセルに対しても、同様にして本発明を適用することは可能である。加えて、上記の実施の形態では、通常のFAシステムを対象としたが、さらに、複数のマニピュレータ、センサ、周辺機器などから構成される知能ロボットシステムに対しても、同様にして本発明を適用することは可能であり、それ故、本発明におけるFAシステムとは、かかる知能ロボットシステムをも含む概念である。

【0082】

【発明の効果】以上の詳細な説明からも明らかなように、本発明によれば、ユーザが、ロボットなどの自動機械と視覚センサなどのプログラミング可能な周辺機器を含む複数機器を組み合わせて、FAシステムを構築する

際に、各機器を制御するプログラムをセル制御プログラムとして一本化して記述することができることはもちろんのこと、さらに、このセル制御プログラムから、FAシステムの制御装置のプロセッサの構成と性能に応じて、各プロセッサの処理負荷の配分が最適となるように、各プロセッサ用のプログラムを自動的に生成することができるので、従来のように制御装置の構成に応じて複数のプログラムを作成したり、幾つものプログラミング言語を習得したりする必要がなくなり、その結果として、FAシステムの制御プログラムの開発効率を向上させることができるという効果を発揮する。

【図面の簡単な説明】

【図1】本発明の特徴点であるFAシステムの制御装置におけるプロセッサ別モジュール分割手段のプロセッサ別プログラム生成の処理プロセスを示す図である。

【図2】本発明の実施の形態になるFAシステムの制御装置における処理プロセスを模式的に示すブロック図である。

【図3】上記図2のセル制御プログラム編集手段における端末の画面表示の一例を示す図である。

【図4】上記本発明を適用した電子部品挿入作業の一部を表すペトリネットの一例を示す図である。

【図5】上記図4の電子部品挿入作業における部品挿入直前の様子を示す図である。

【図6】上記図4の電子部品挿入作業における部品挿入直後の様子を示す図である。

【図7】上記図4のペトリネットをコード化したセル制御プログラムのcellブロックを示す図である。

【図8】上記図4のペトリネットをコード化したセル制御プログラムのdefブロックを示す図である。

【図9】上記図2の機能別モジュール分離変換手段において、機能別プログラムモジュールを生成する処理プロセスを示すブロック図である。

【図10】上記図7及び図8に示したセル制御プログラムから生成したシーケンス制御プログラムの一部（ROB1のユニットブロック）を示す図である。

【図11】上記図7及び図8に示したセル制御プログラムから生成したシーケンス制御プログラムの一部（ROB2のユニットブロック）を示す図である。

【図12】上記図7及び図8に示したセル制御プログラムから生成したシーケンス制御プログラムの一部（CAM1及びCAM2のユニットブロック）を示す図である。

【図13】上記図7及び図8に示したセル制御プログラムから生成したロボット制御プログラムの一例を示す図である。

【図14】上記図7及び図8に示したセル制御プログラムから生成した画像処理プログラムの一例を示す図である。

【図15】上記図9に示す機能別モジュール分離変換手

段の処理の概略を示すフローチャートである。

【図16】上記図15に示すフローの機能別データ抽出処理の詳細を示すフローチャートである。

【図17】上記図16に示すフローの1プレイス毎のネット構造解析処理の詳細を示すフローチャートの前半部分である。

【図18】上記図16に示すフローの1プレイス毎のネット構造解析処理の詳細を示すフローチャートの後半部分である。

【図19】上記図16に示すフローの1プレイス毎のコマンド列解析処理の詳細を示すフローチャートである。

【図20】上記図16に示すフローの1プレイス毎の終了条件解析処理の詳細を示すフローチャートである。

【図21】上記図15に示すフローのシーケンス制御プログラム生成処理の詳細を示すフローチャートの前半部分である。

【図22】上記図15に示すフローシーケンス制御プログラム生成処理の詳細を示すフローチャートの後半部分である。

【図23】上記図15に示すフローのロボット制御プログラム生成処理の詳細を示すフローチャートである。

【図24】上記図15に示すフローの画像処理プログラム生成処理の詳細を示すフローチャートである。

【図25】上記図9に示した機能別モジュール分離変換手段によって、セル制御プログラムからシーケンス制御プログラムとロボット制御プログラムを生成する処理プロセスの一例を示す図である。

【図26】上記図9に示した機能別モジュール分離変換手段によって、セル制御プログラムからシーケンス制御プログラムと画像処理プログラムを生成する処理プロセスの一例を示す図である。

【図27】上記図1に示した機能別プログラムモジュールのブロック構成を示す図である。

【図28】上記図1に示した機能別プログラムモジュールのコード実行手順を示すフロー図である。

【図29】上記図2に示した機能別プログラムモジュールからプロセッサ別プログラム群を生成する処理プロセスの一例を示す図である。

【図30】本発明を適用した一実施の形態になる制御装置（マルチプロセッサ型コントローラ）の構成を示すブロック図である。

【図31】上記図30に示した制御装置における共有データの構成を示す図である。

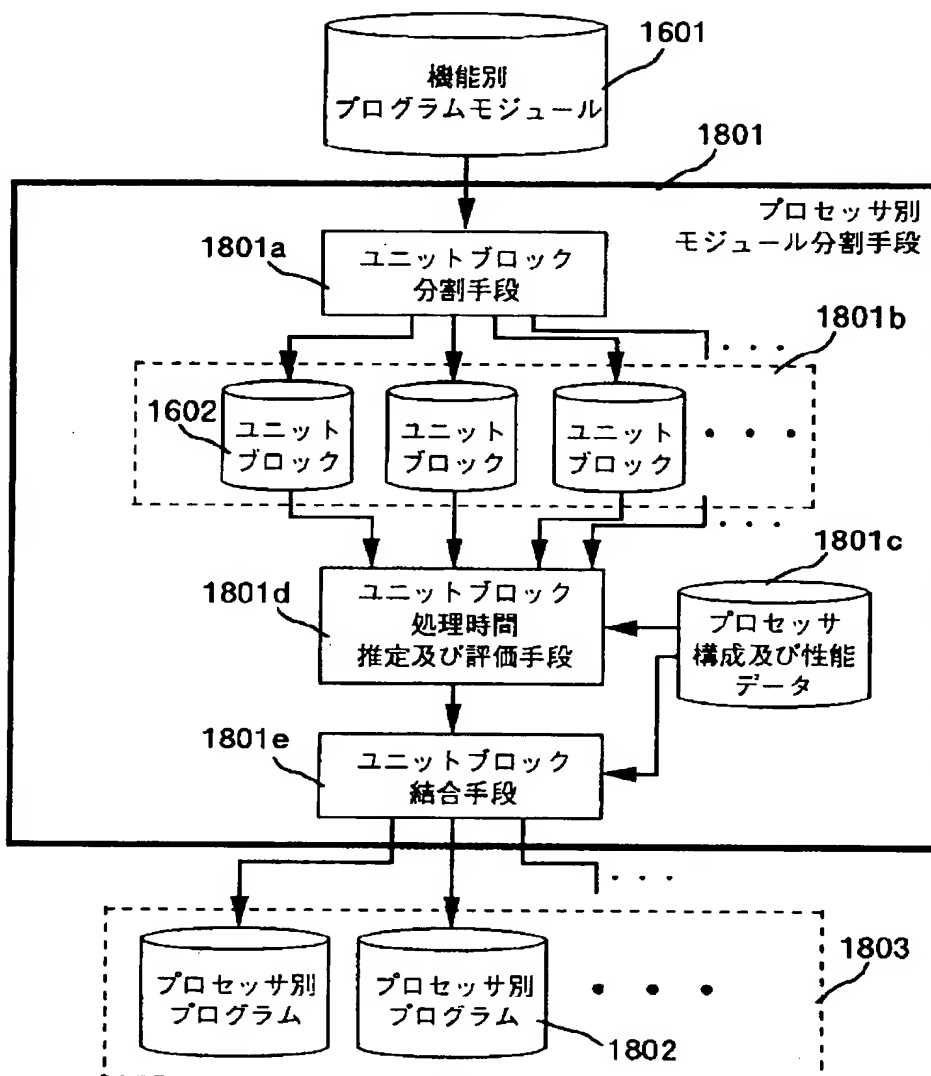
【図32】本発明を適用した他の実施の形態になる制御装置（従来型のコントローラを組み合わせた制御システム）の構成を示す図である。

【図33】本発明を適用したさらに他の実施の形態になる制御装置（マルチプロセッサ型コントローラを複数個組み合わせた制御システム）の構成を示す図である。

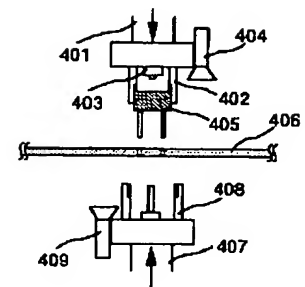
【符号の説明】

- 101 セル制御プログラム編集手段
- 102 セル制御プログラム
- 103 機能別モジュール分離変換手段
- 103a 機能別データ抽出手段
- 103b ネット構造
- 103c 状態終了条件
- 103d I/O制御命令列
- 103e ロボット制御命令列
- 103f ロボット動作タイミング
- 103g 画像処理命令列
- 103h 画像処理タイミング
- 103i シーケンス制御プログラム生成手段
- 103j ロボット制御プログラム生成手段
- 103k 画像処理プログラム生成手段
- 104 シーケンス制御プログラムモジュール
- 105 ロボット制御プログラムモジュール
- 106 画像処理プログラムモジュール
- 107 シーケンス制御プロセッサ別モジュール分割手段
- 108 ロボット制御プロセッサ別モジュール分割手段
- 109 画像処理プロセッサ別モジュール分割手段
- 110 シーケンス制御プロセッサ用プログラム群
- 111 ロボット制御プロセッサ用プログラム群
- 112 画像処理プロセッサ用プログラム群
- 113 シーケンス制御プロセッサ群
- 114 ロボット制御プロセッサ群
- 115 画像処理プロセッサ群
- 116 セル
- 116a、116b ロボット
- 116c、116d 視覚センサ
- 116e、116f 周辺機器
- 1601 機能別プログラムモジュール
- 1801 プロセッサ別モジュール分割手段
- 1801a ユニットブロック分割手段
- 1801b ユニットブロック群
- 1801c プロセッサ構成及び性能データ
- 1801d ユニットブロック処理時間推定及び結合手段
- 1801e ユニットブロック結合手段
- 1802 プロセッサ別プログラム
- 1803 プロセッサ別プログラム群

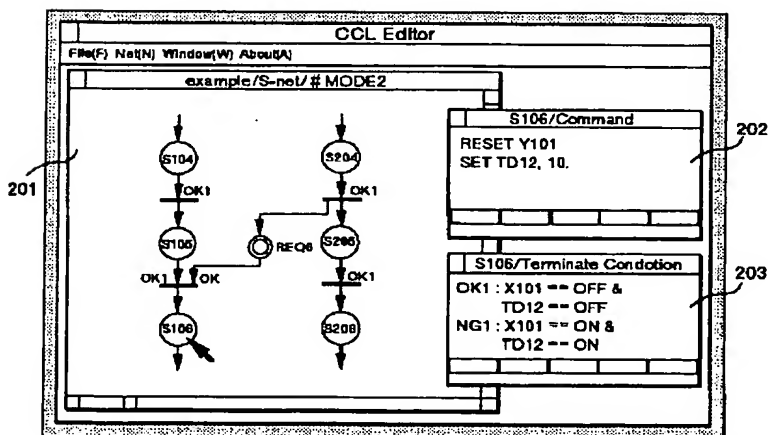
【図1】



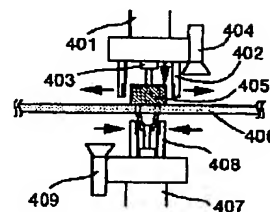
【図5】



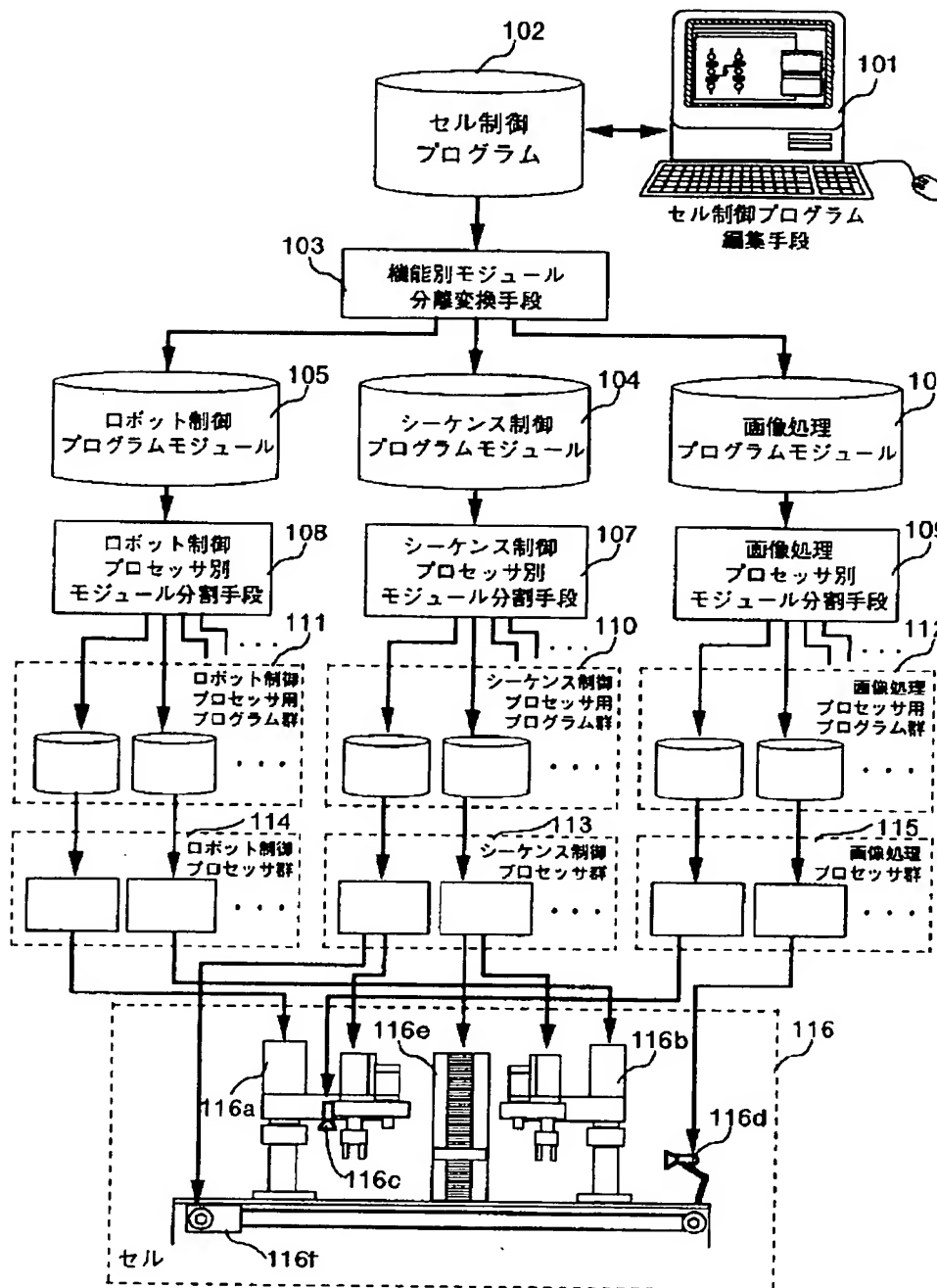
【図3】



【図6】



【図2】



【図12】

#CAM1

```
IF S150 == CompleteNormally
& REQ1 == CompleteNormally
->
```

```
TERM S150
TERM REQ1
EXEC S151
```

```
IF S151 == CompleteNormally
->
```

```
TERM S151
EXEC S150
EXEC REQ2
```

#CAM2

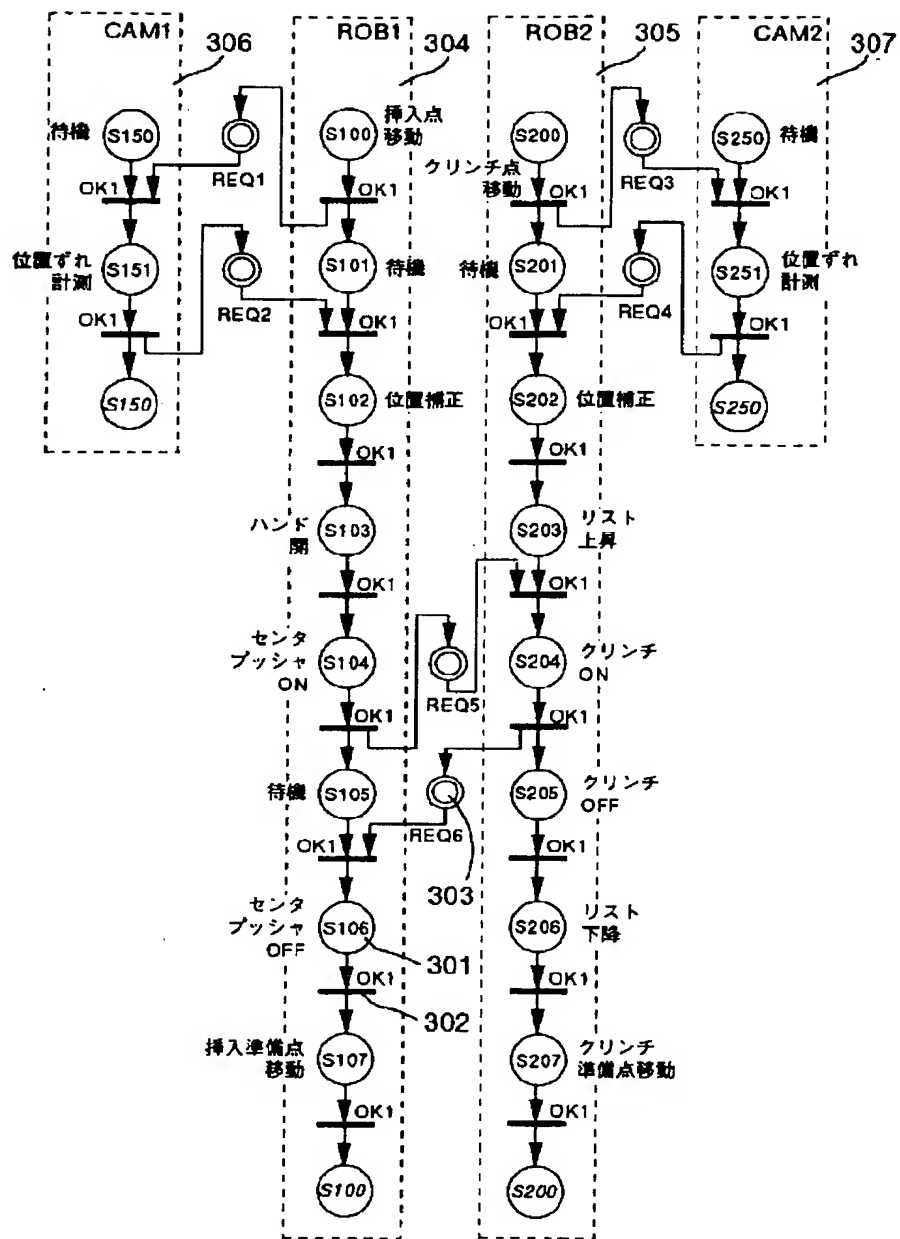
```
IF S250 == CompleteNormally
& REQ3 == CompleteNormally
->
```

```
TERM S250
TERM REQ3
EXEC S251
```

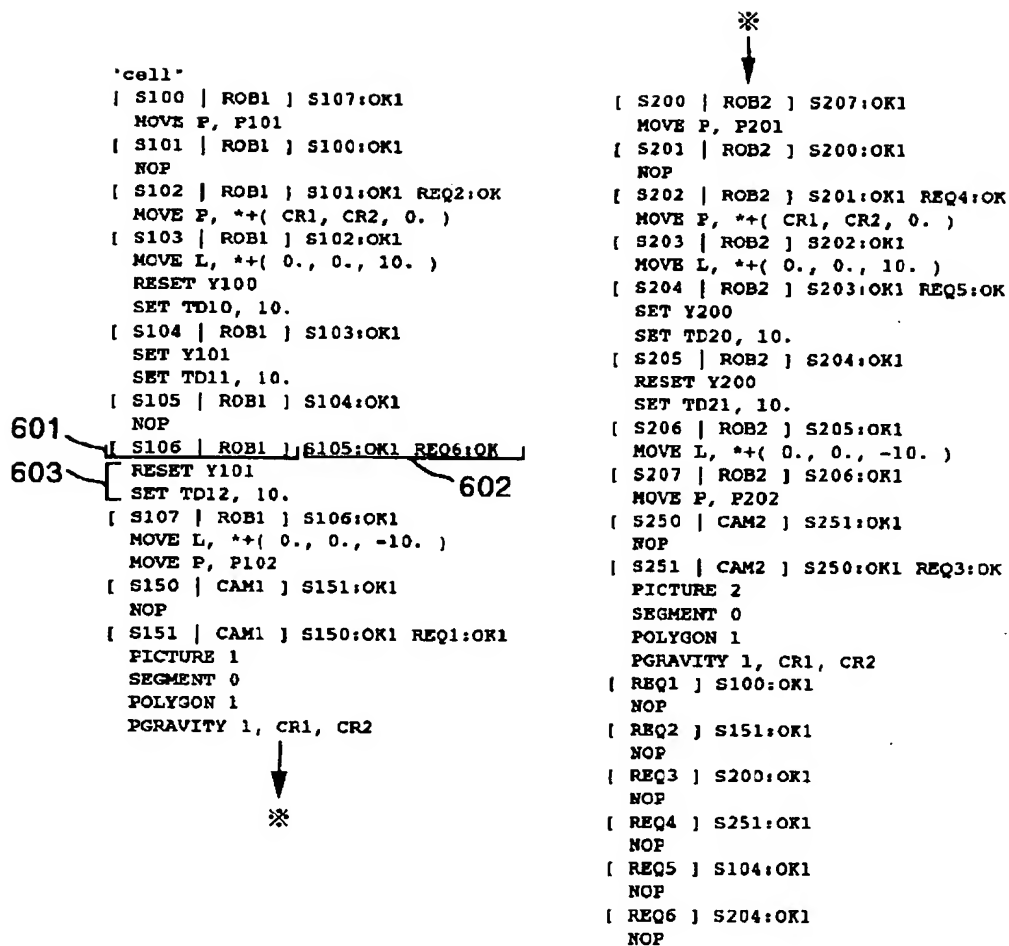
```
IF S251 == CompleteNormally
->
```

```
TERM S251
EXEC S250
EXEC REQ4
```

【図4】



【図7】



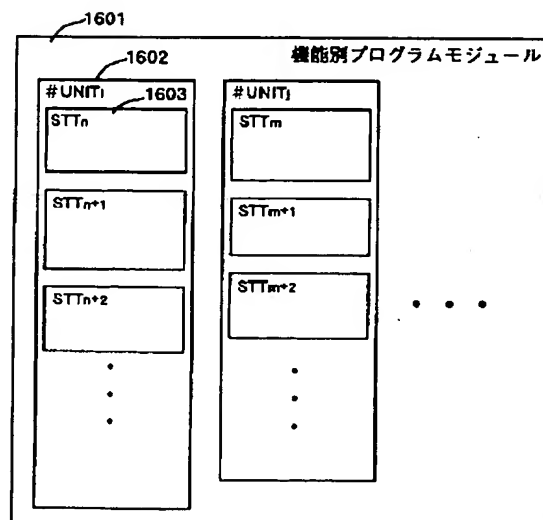
【図14】

```

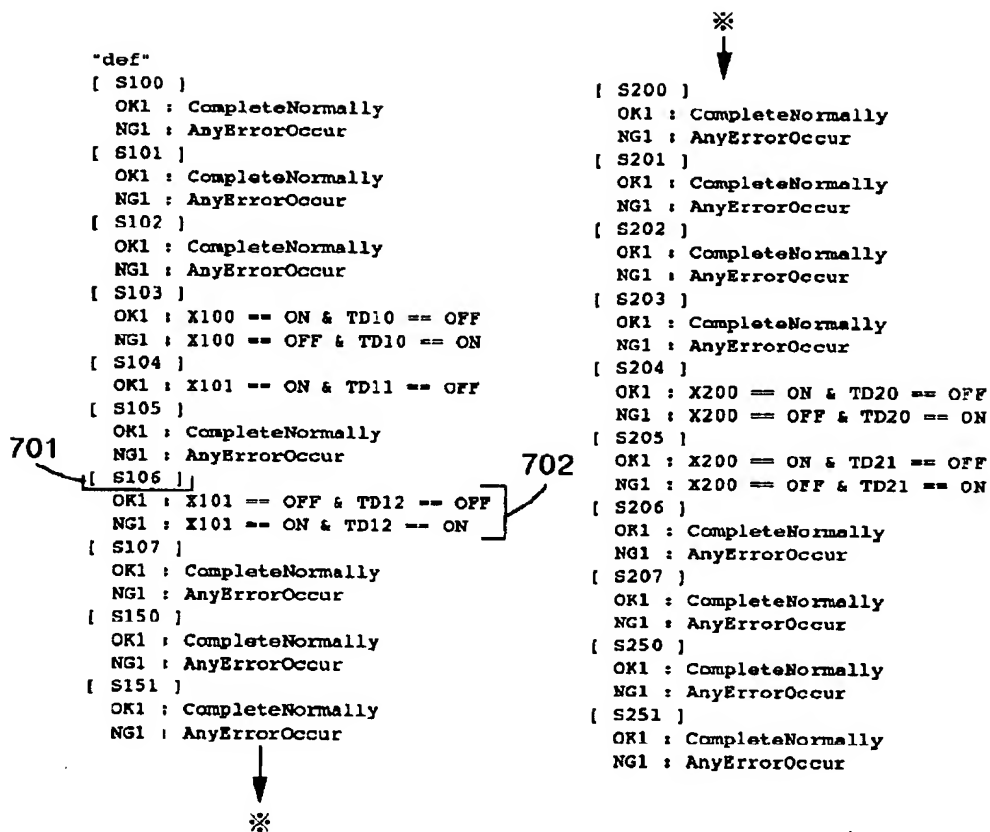
#CAM1
STT150
  NOP
STT151
  PICTURE 1
  SEGMENT 0
  POLYGON 1
  PGRAVITY 1, CR1, CR2

#CAM2
STT250
  NOP
STT251
  PICTURE 2
  SEGMENT 0
  POLYGON 1
  PGRAVITY 1, CR1, CR2
  
```

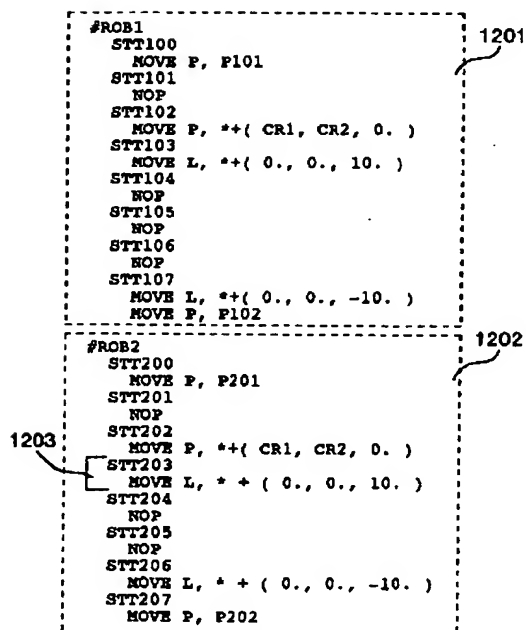
【図27】



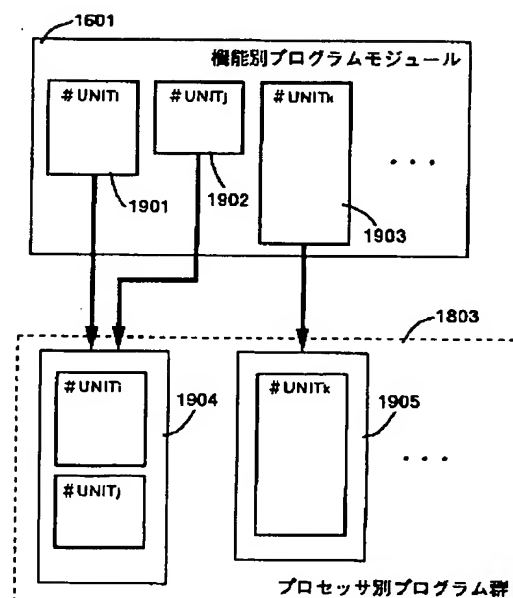
【図8】



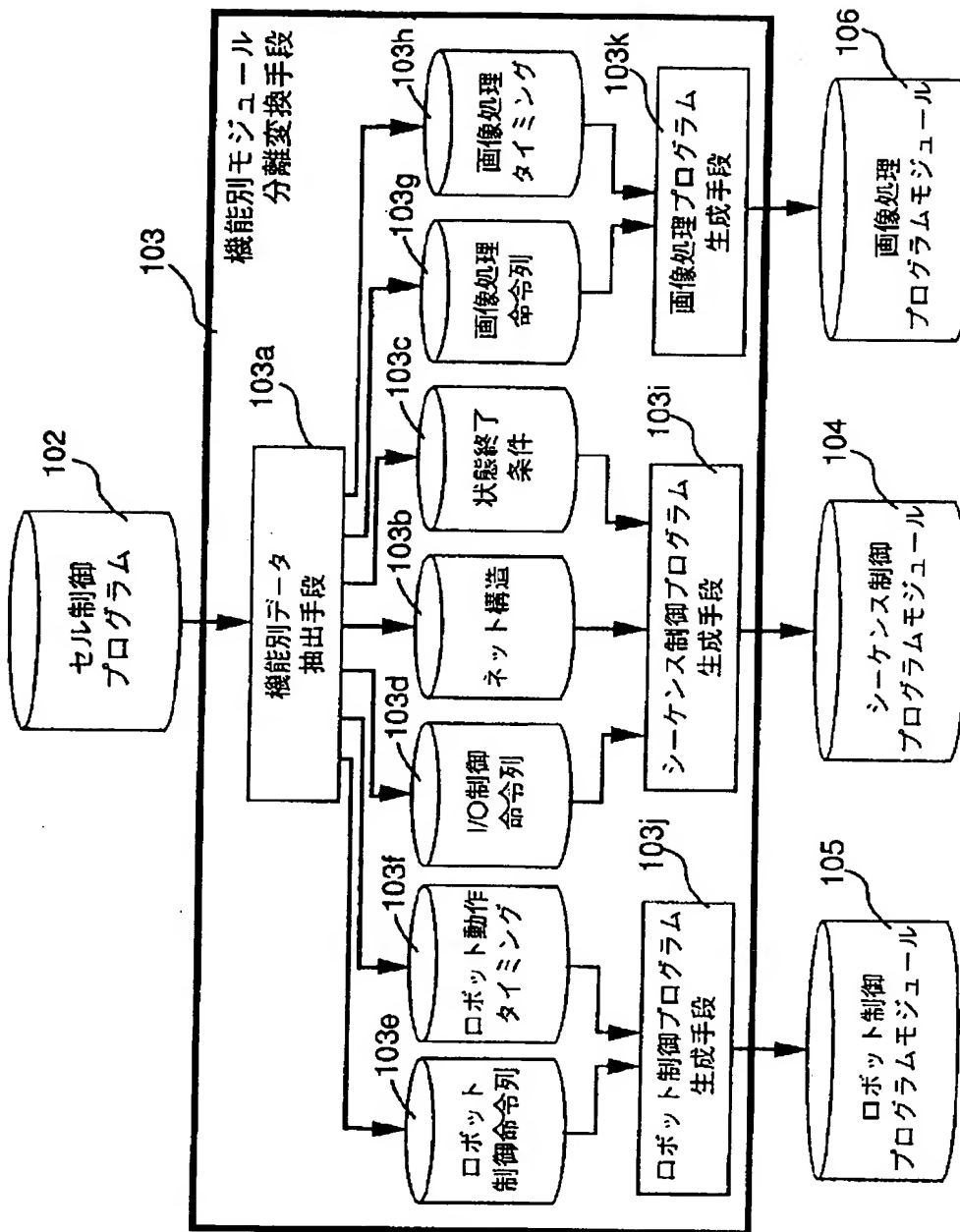
【図13】



【図29】



【図9】



【図10】

```

※
┆
#ROB1
IF S100 == CompleteNormally
->
  TERM S100
  EXEC S101
  EXEC REQ1

IF S101 == CompleteNormally
& REQ2 == CompleteNormally
->
  TERM S101
  TERM REQ6
  EXEC S102

IF S102 == CompleteNormally
->
  TERM S102
  EXEC S103
  RESET Y100
  SET TD10 , 10.

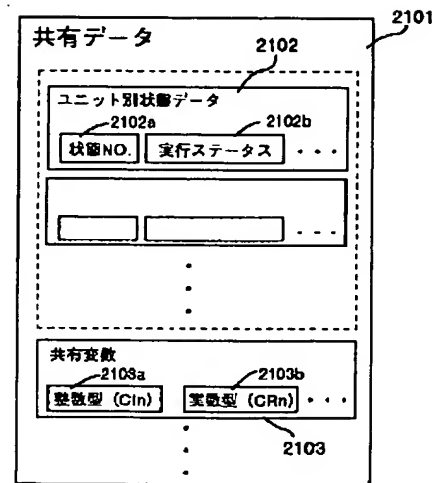
IF S103 == Executing
& X100 == ON
& TD10 == OFF
->
  TERM S103
  EXEC S104
  SET Y101
  SET TD11 , 10.
  ※
  IF S104 == Executing
  & X101 == ON
  & TD11 == OFF
  ->
    TERM S104
    EXEC S105
    EXEC REQ5

  IF S105 == CompleteNormally
  & REQ6 == CompleteNormally
  ->
    TERM S105
    TERM REQ6
    EXEC S106
    RESET Y101
    SET TD12 , 10.

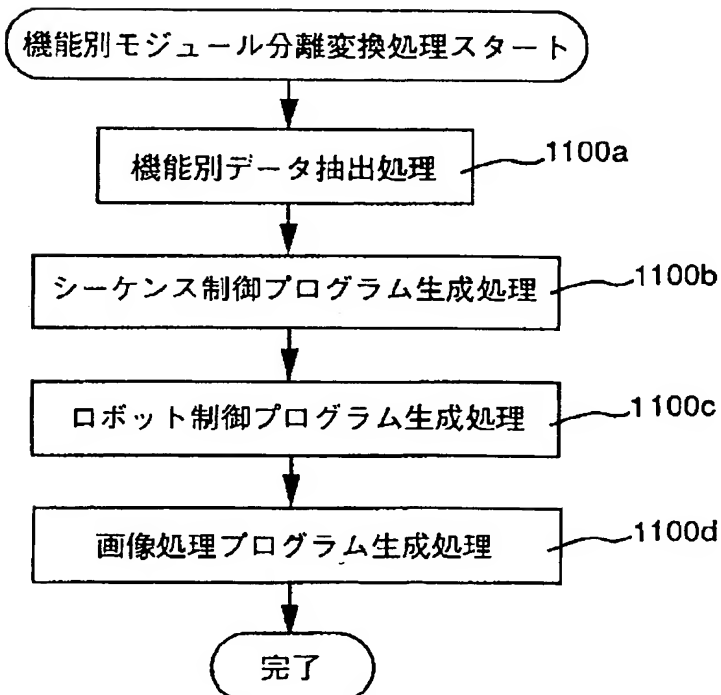
  IF S106 == Executing
  & X101 == OFF
  & TD12 == OFF
  ->
    TERM S106
    EXEC S107

  IF S107 == CompleteNormally
  ->
    TERM S107
    EXEC S100
  
```

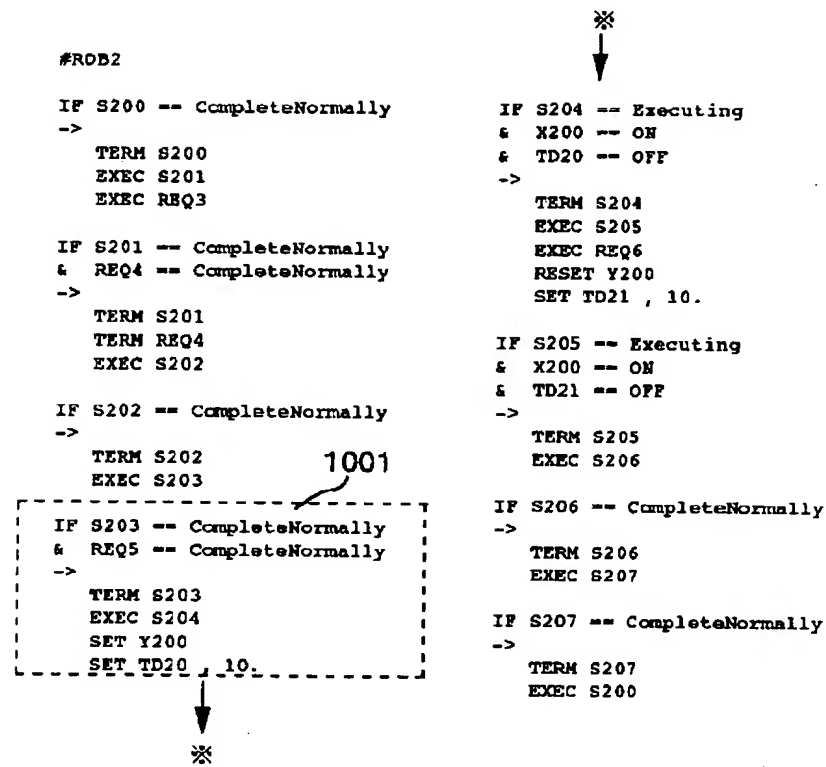
【図31】



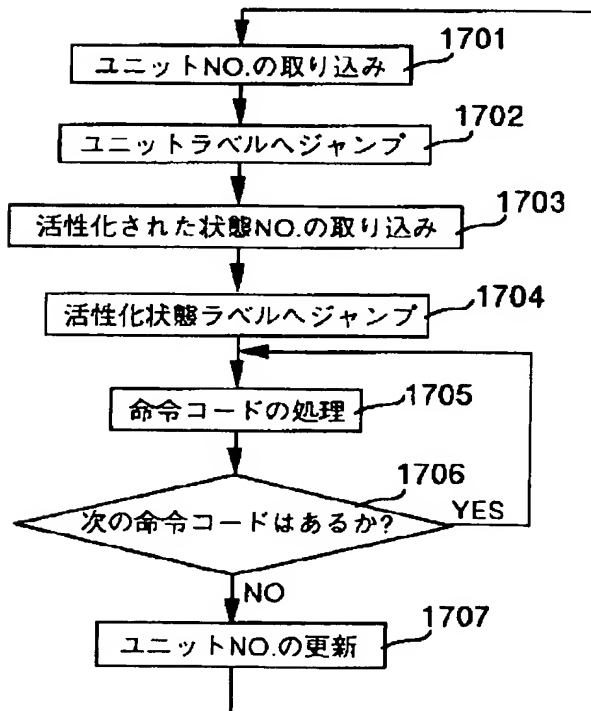
【図15】



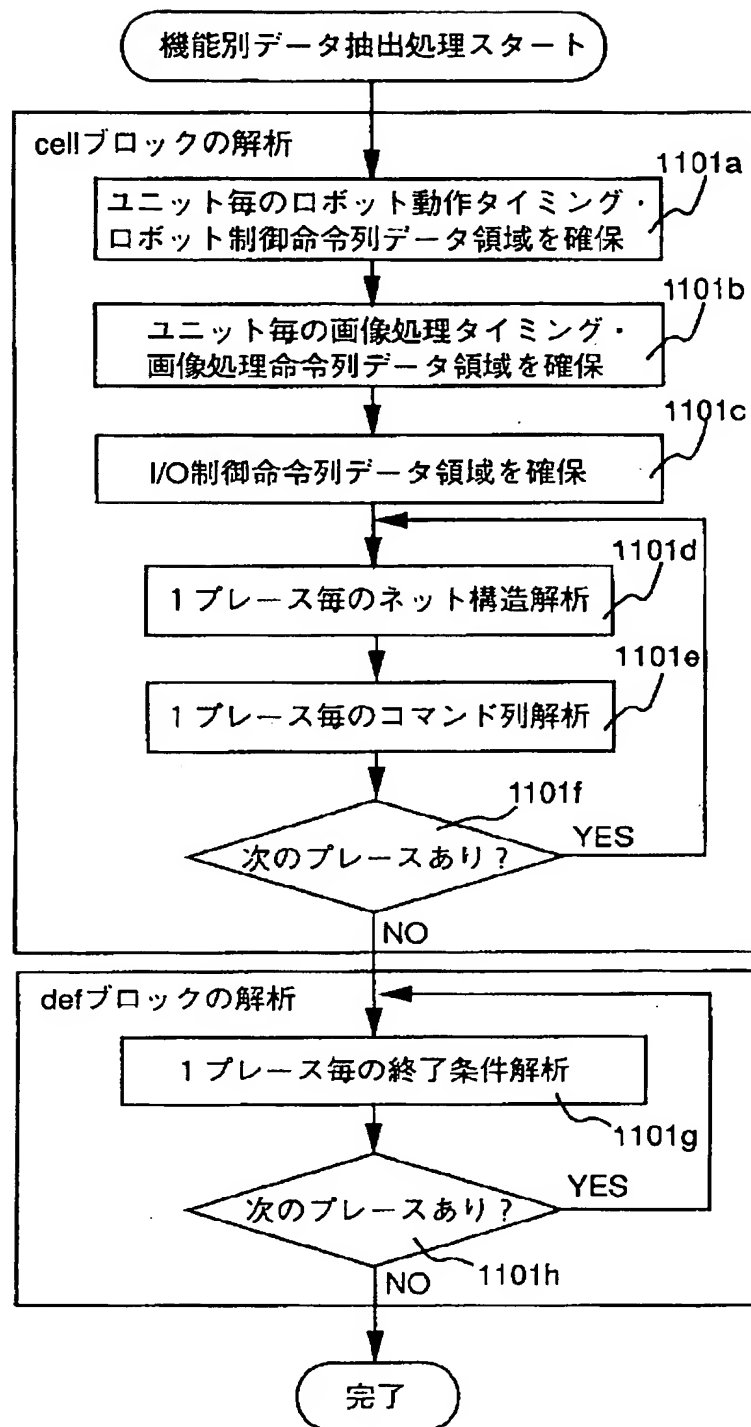
【図11】



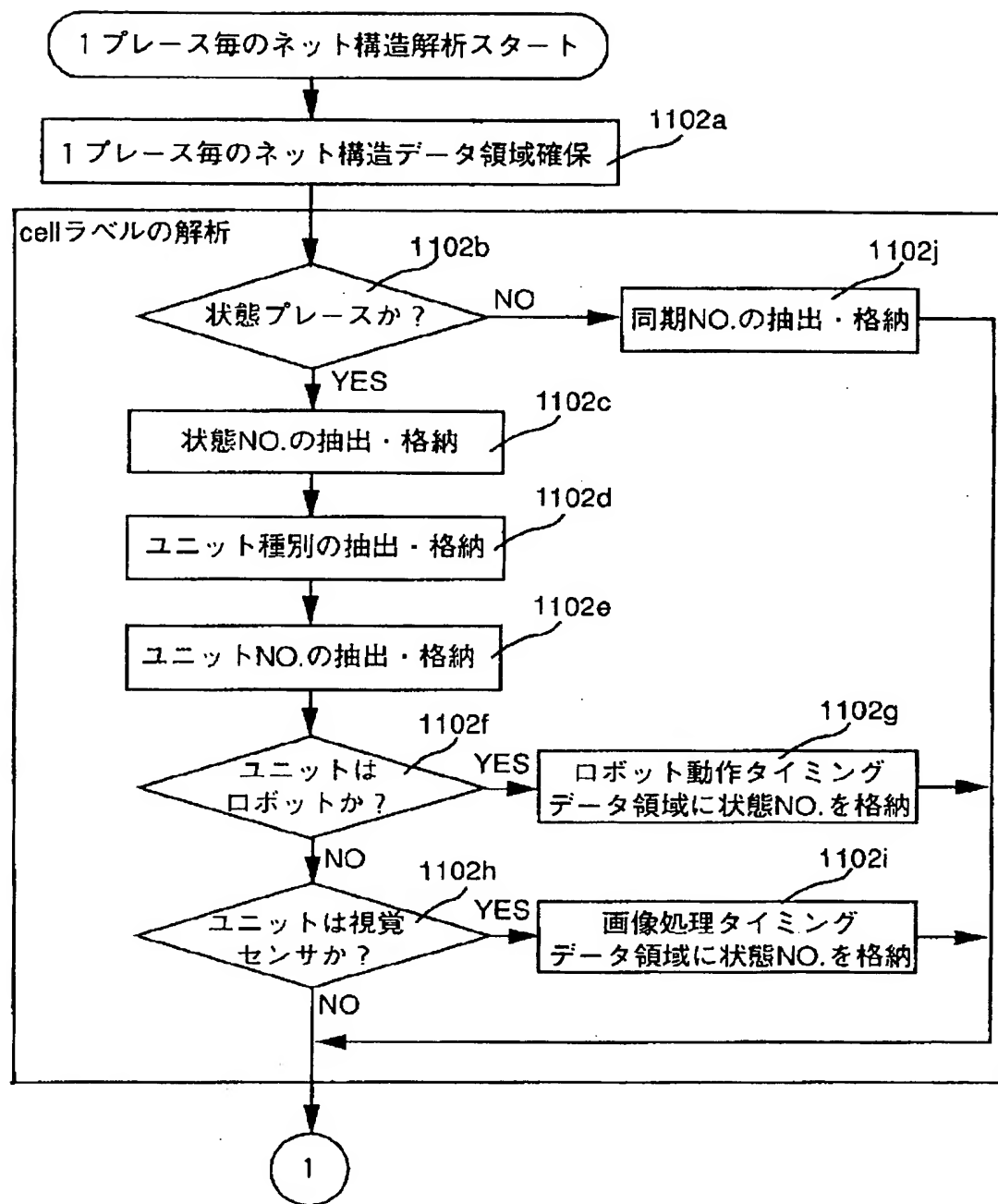
【図28】



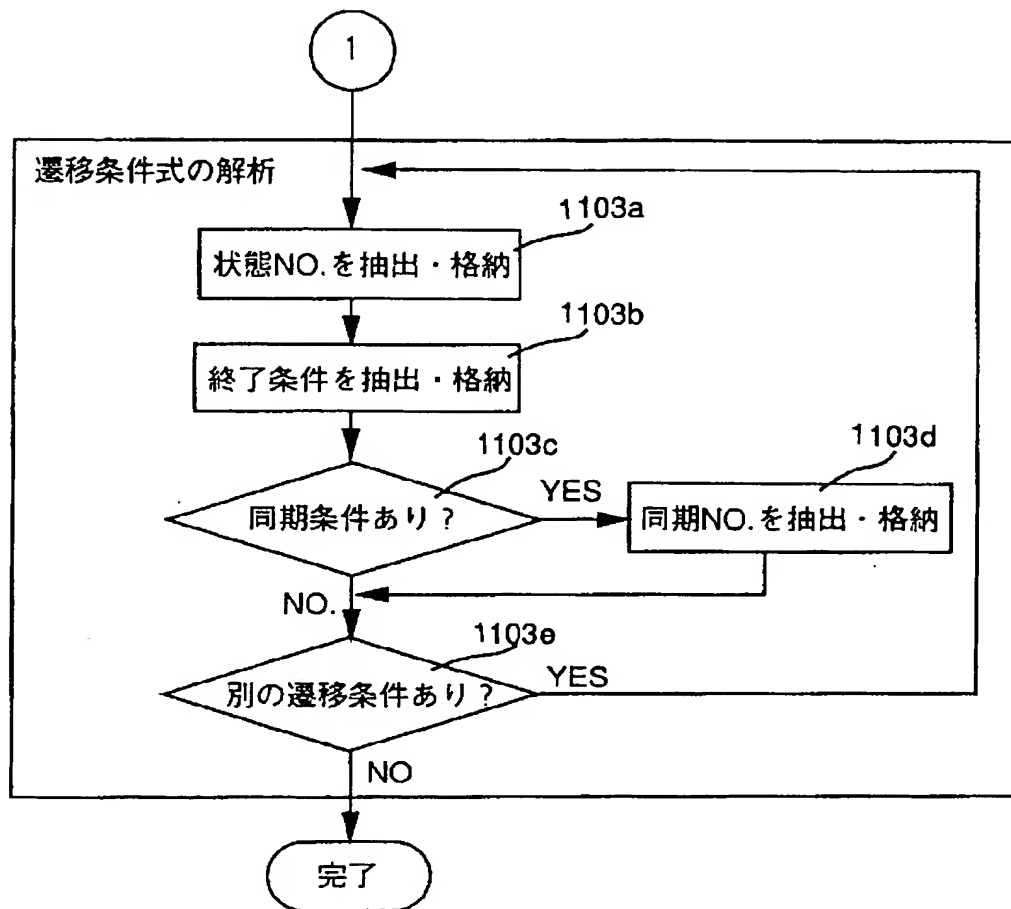
【図16】



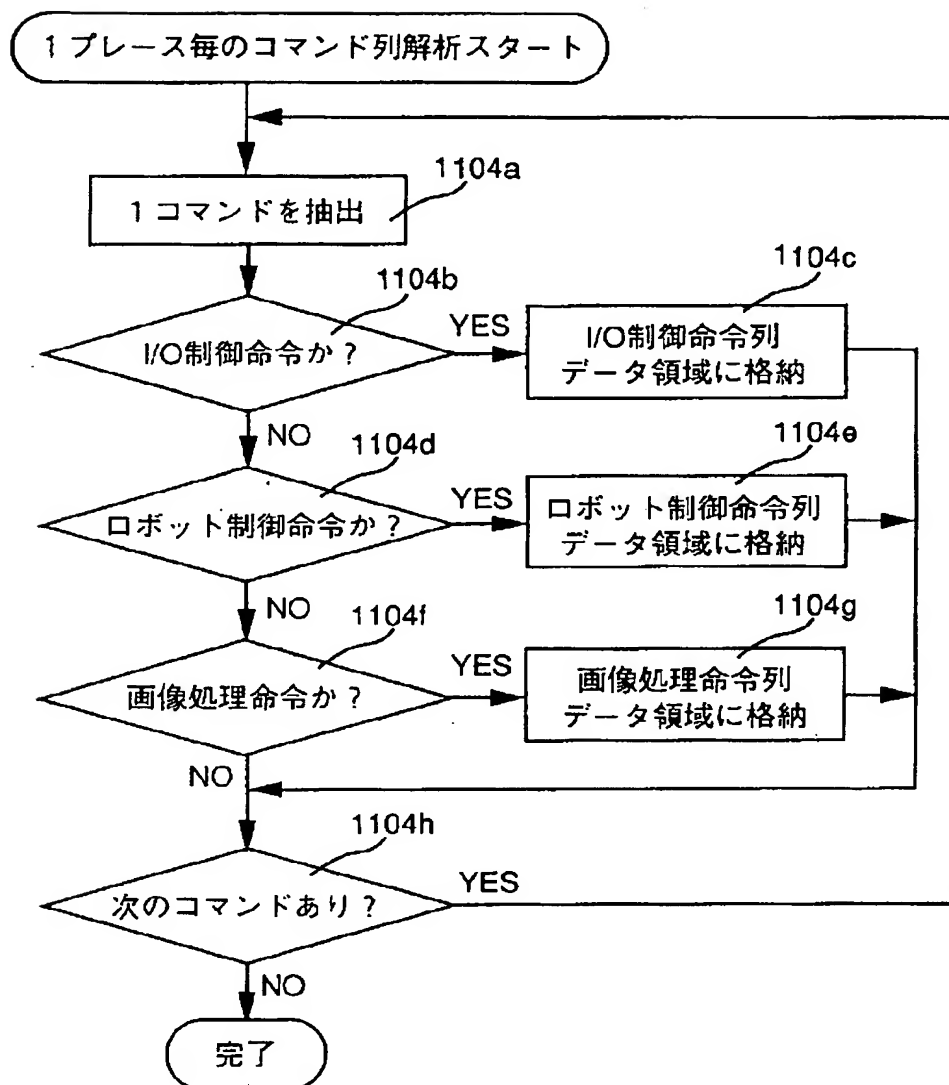
【図17】



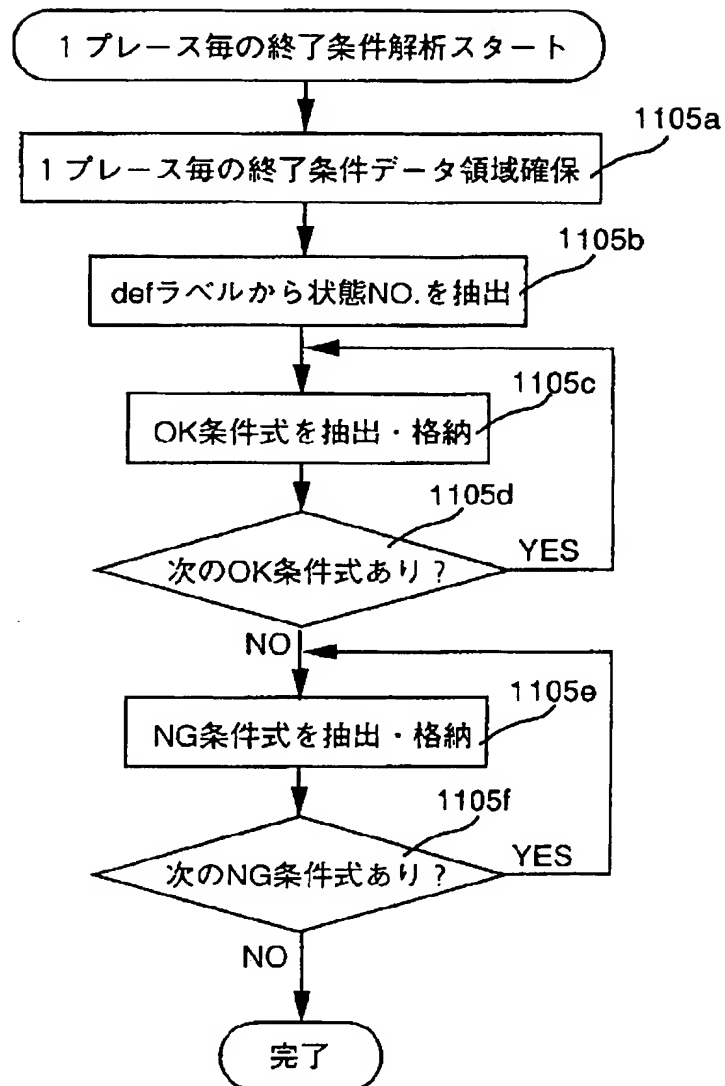
【図18】



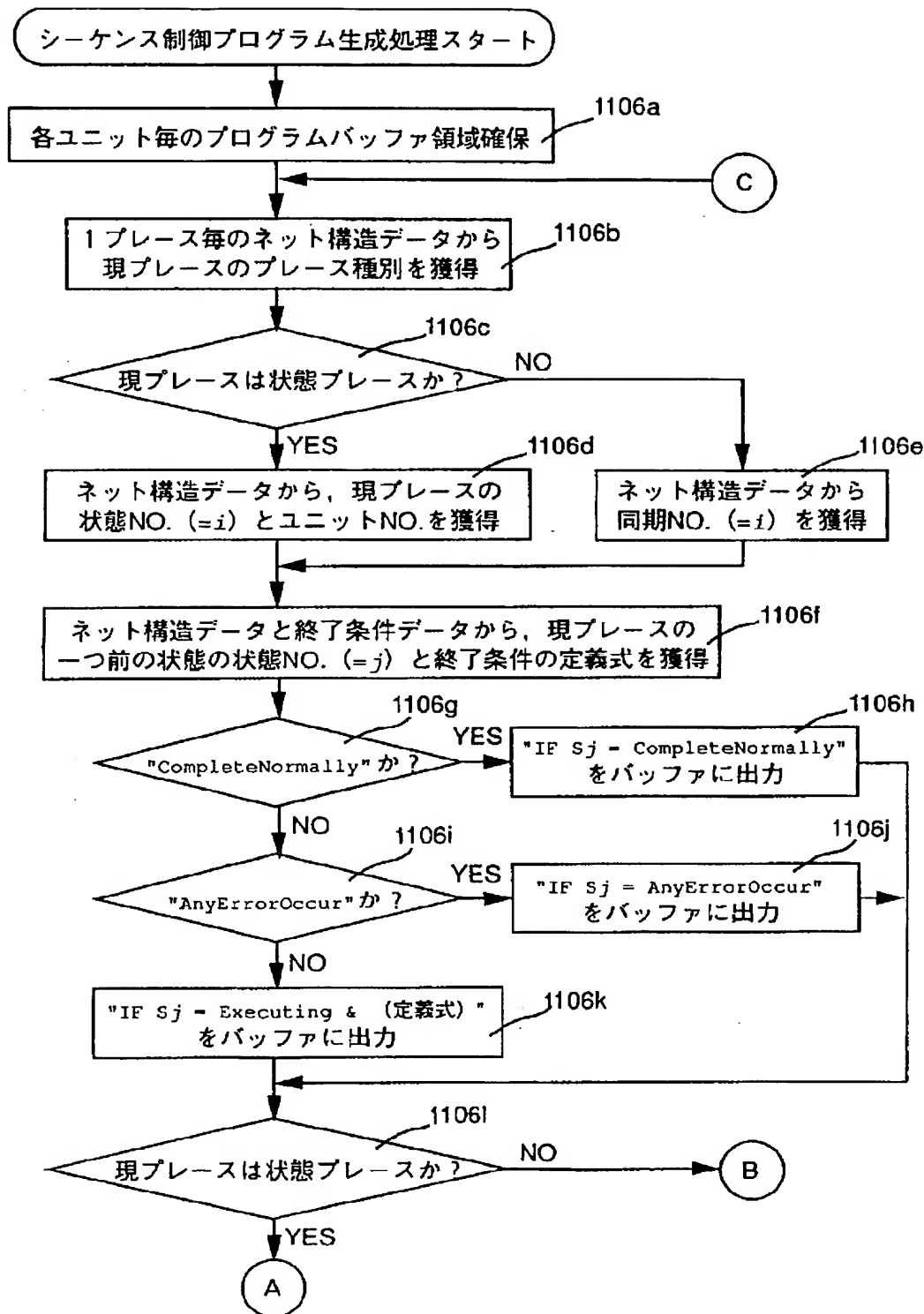
【図19】



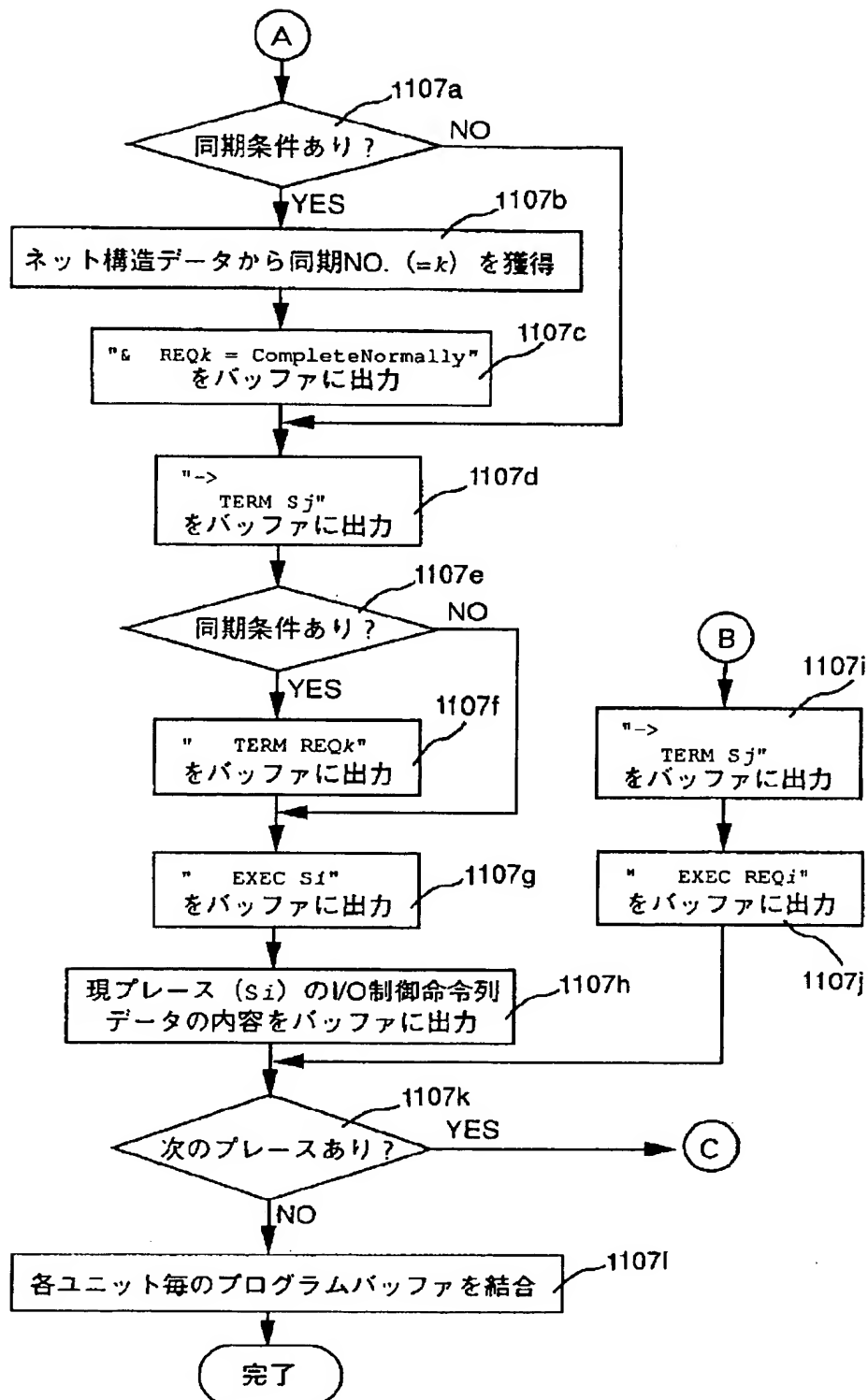
【図20】



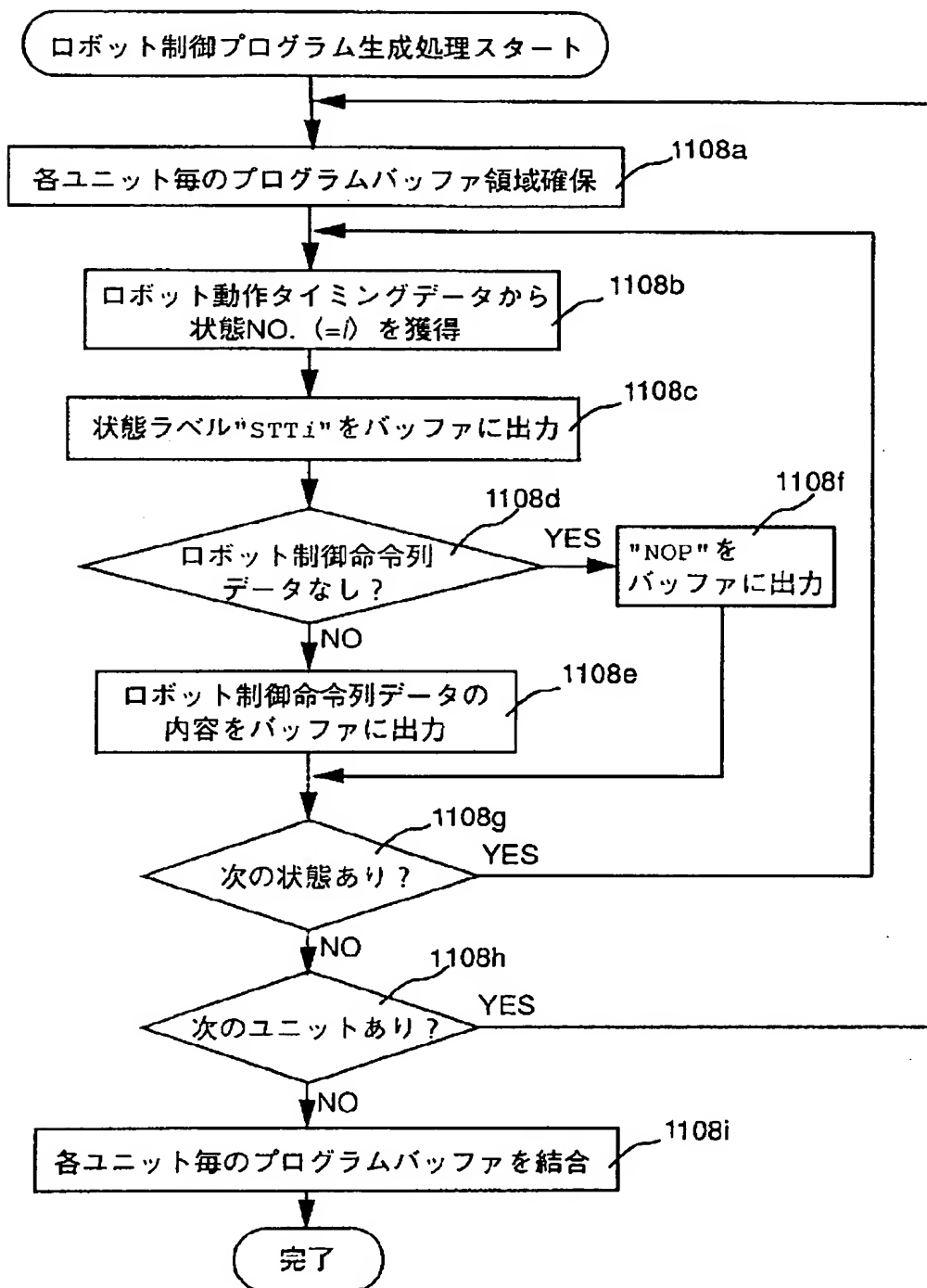
【図21】



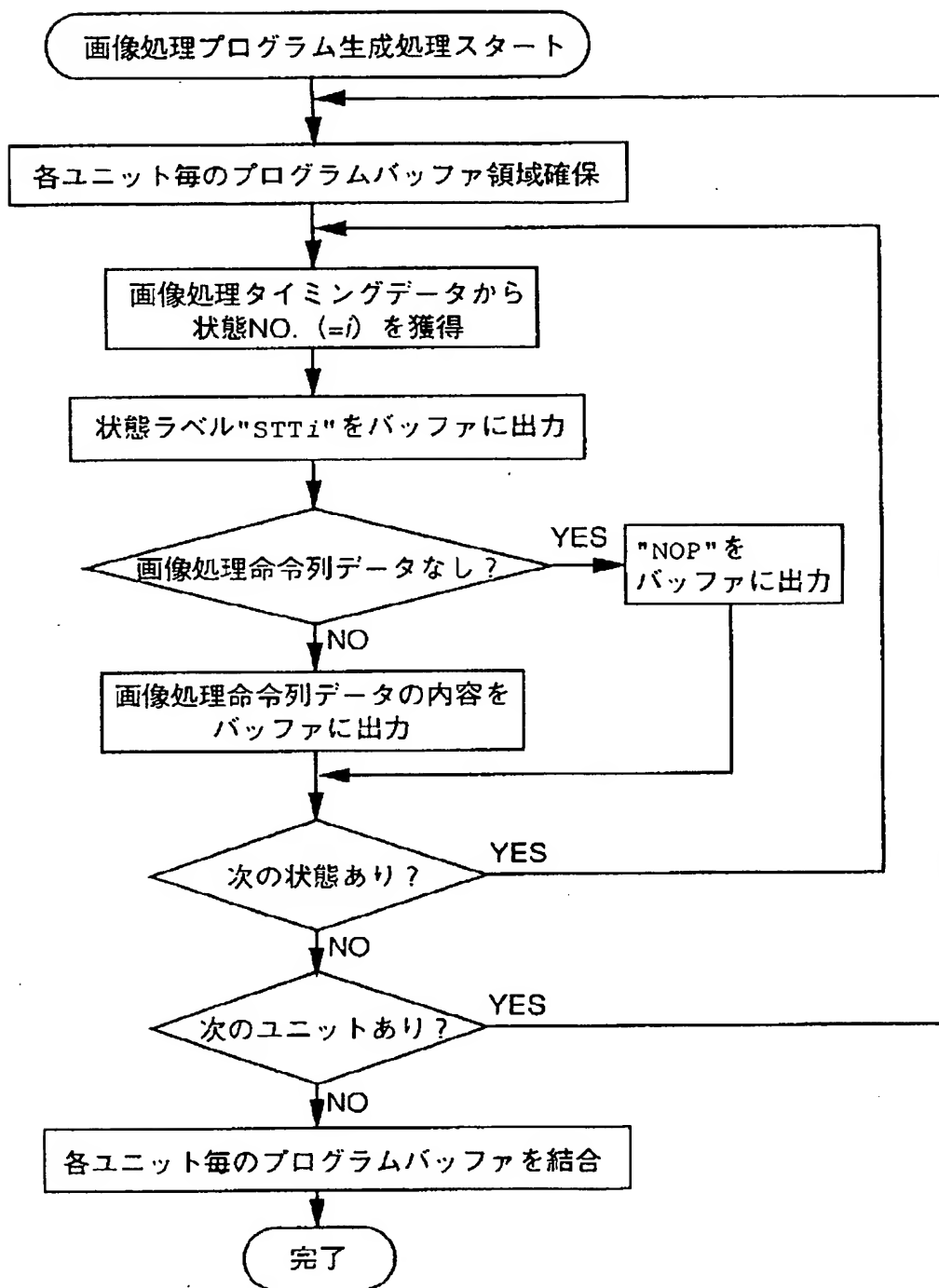
【図22】



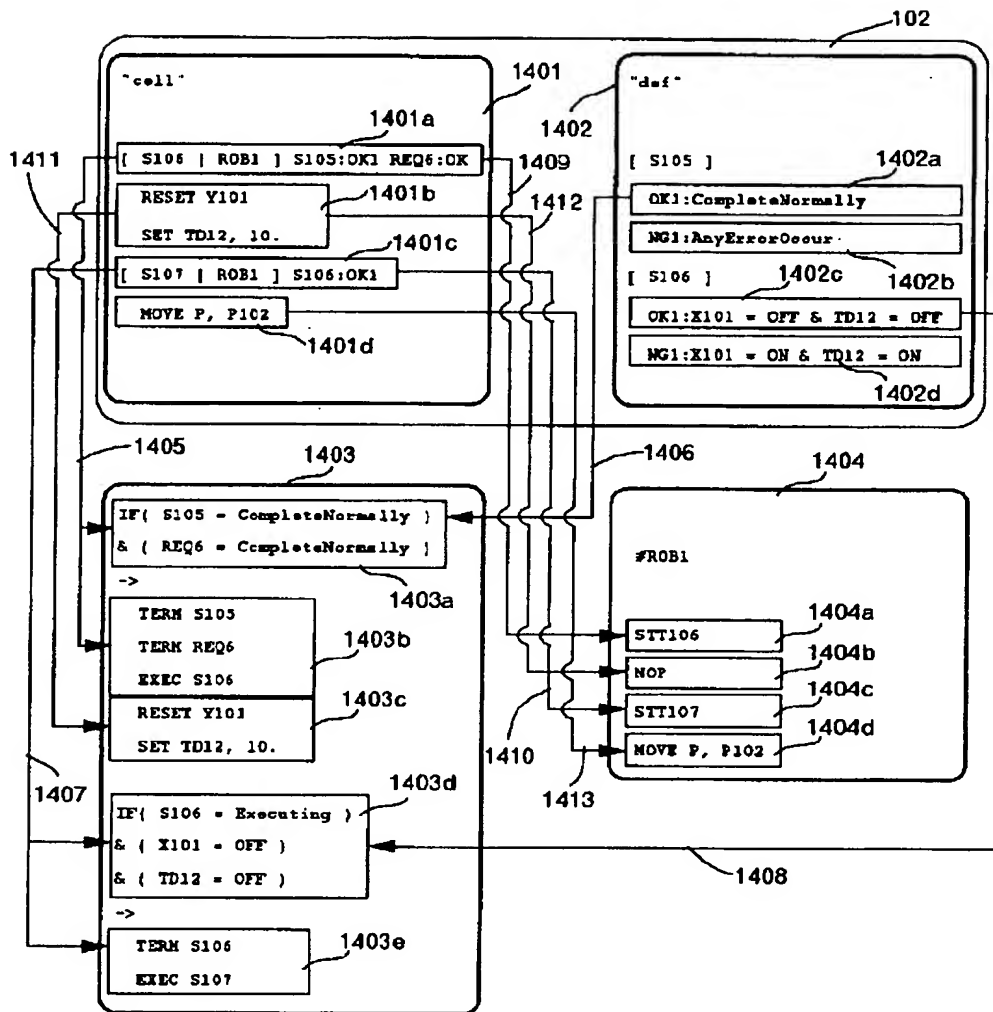
【図23】



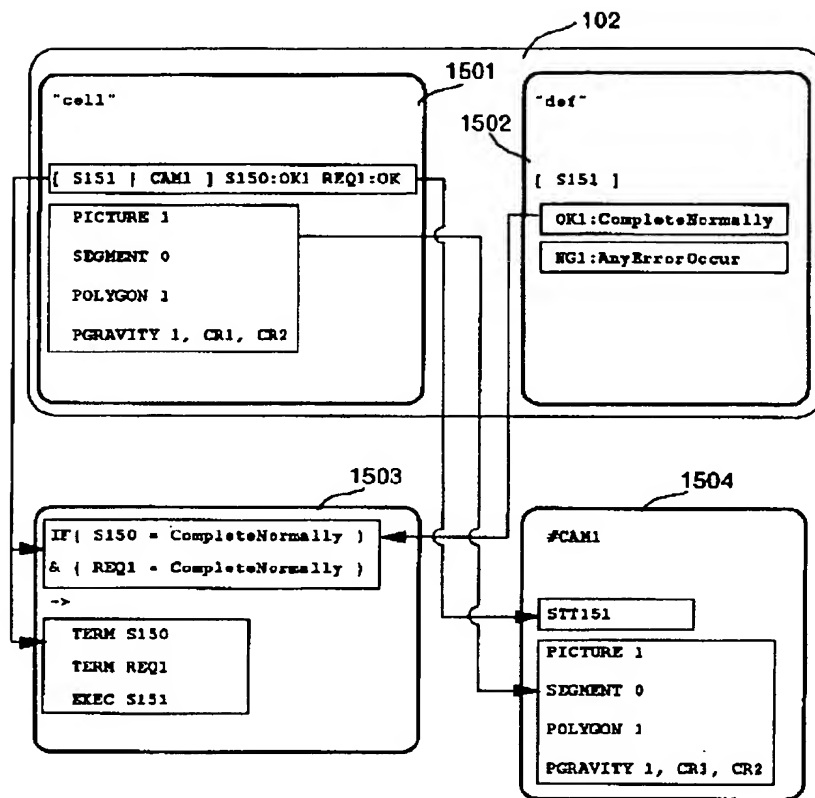
【図24】



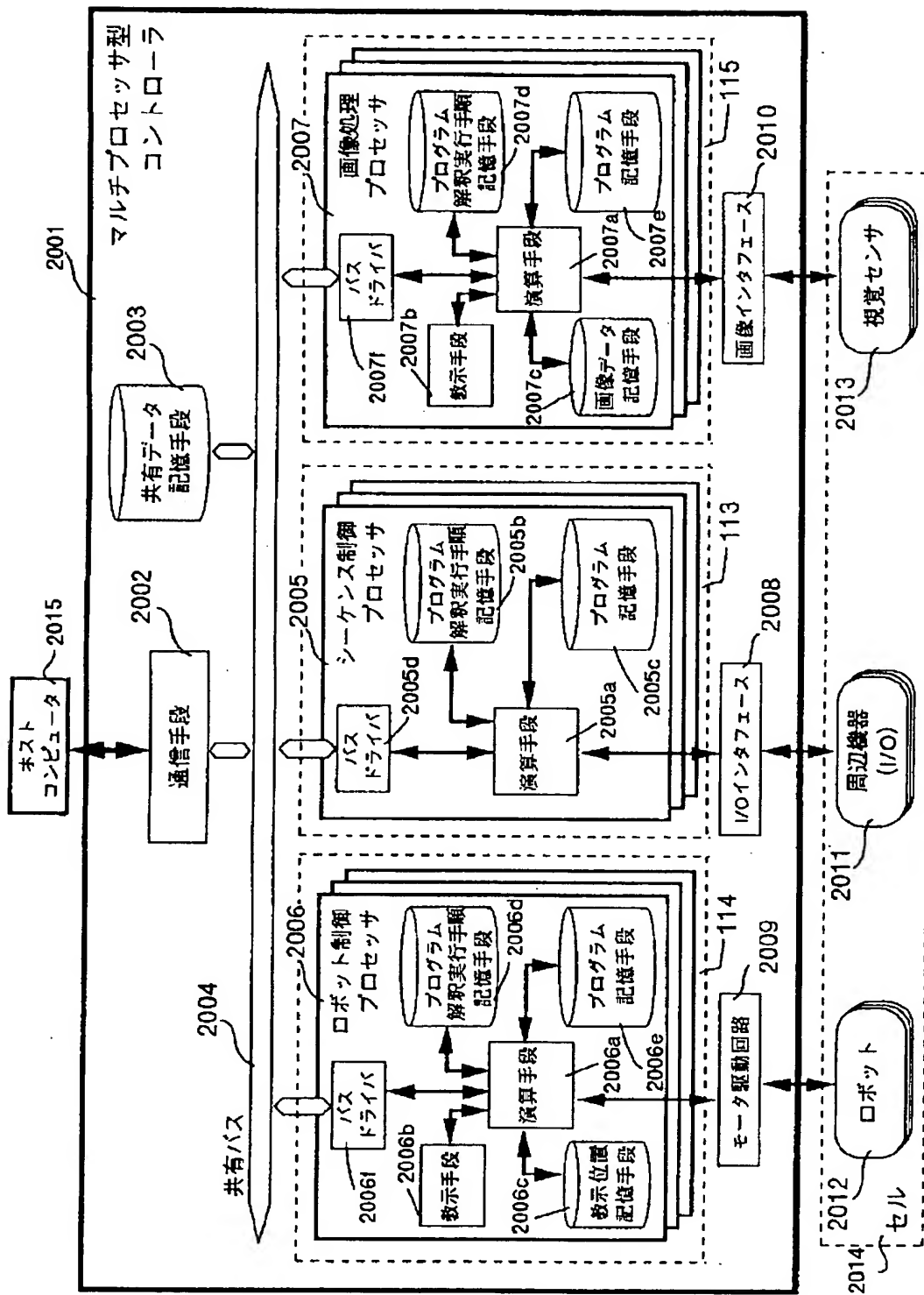
【図25】



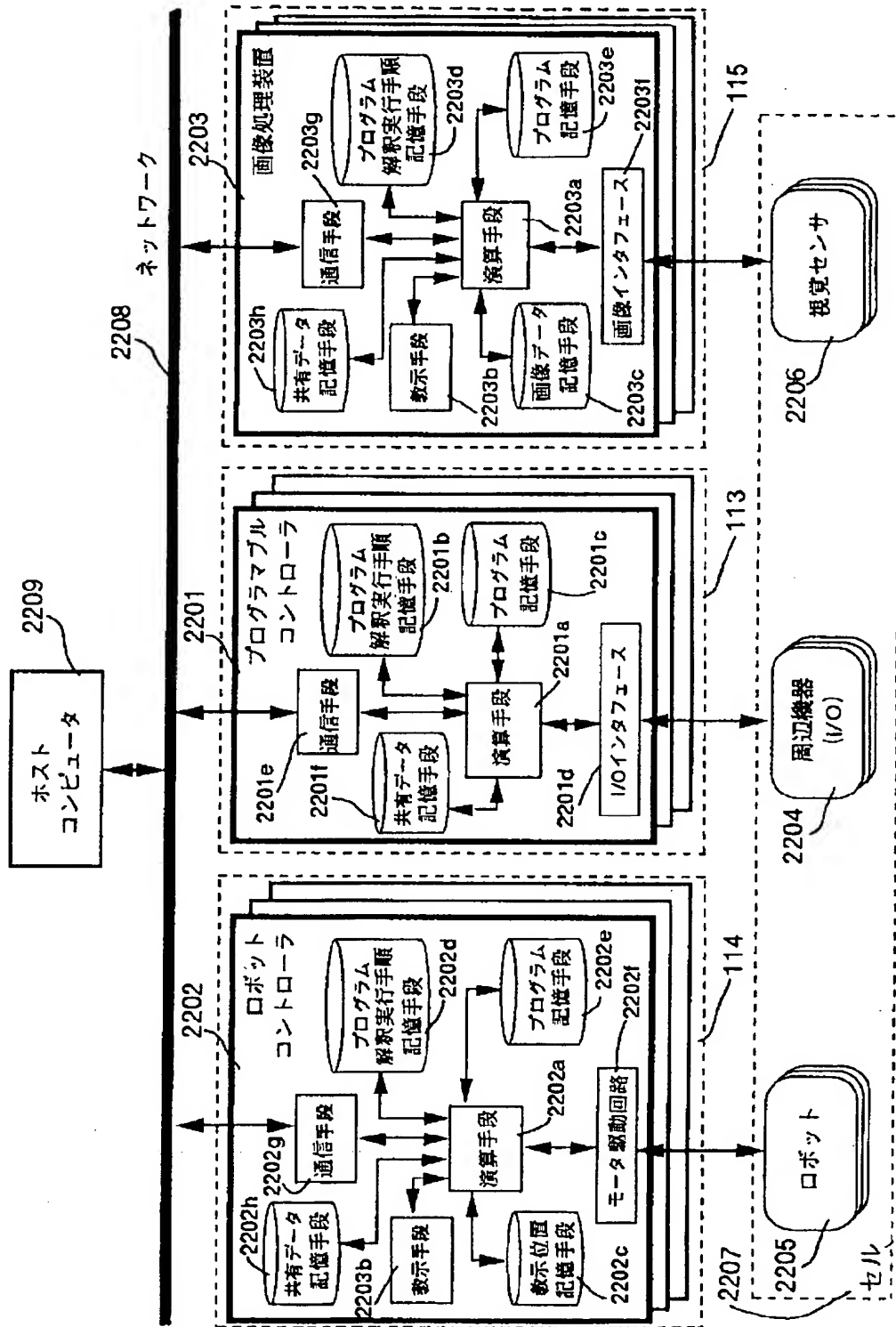
【図26】



【図30】



【図32】



【図33】

